

**An Interactive Multimedia
Continuously Learning Helpdesk System
(When Hal Met SALLY)**

by
Marion L. Groh

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

May 21, 1999

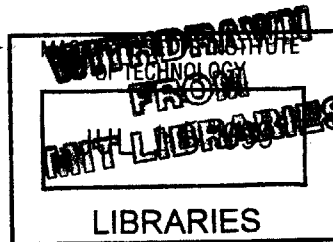
© Copyright 1999 Marion L. Groh. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author _____
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by _____
Patrick H. Winston
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses



ENG

An Interactive Multimedia
Continuously Learning Helpdesk System
(When Hal Met SALLY¹)

by
Marion L. Groh

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 1999

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

SALLY is a helpdesk system implemented in The Intelligent Room. It is an incrementally learning system, which automatically expands its own knowledge. The natural language model of SALLY is simple – a combination of a parsing script and a regular grammar. In addition, SALLY provides help in multimedia form: spoken answers, displayed information, actions.

Thesis supervisor: Patrick H. Winston
Title: Ford Professor, MIT Artificial Intelligence Laboratory

¹ In reference to the 1989 Award-Winning Comedy “*When Harry met Sally*” (Director: Rob Reiner; Writing Credits: Nora Ephron; Distributed by Columbia Pictures)

TABLE OF CONTENTS

1	A NEW KIND OF HELP	6
1.1	OVERVIEW.....	6
1.2	CONTRIBUTIONS	7
1.3	GOALS AND PRINCIPLES.....	7
1.3.1	<i>Scenario</i>	7
1.3.2	<i>Categories of Help</i>	10
1.3.3	<i>Not A Manual</i>	11
1.3.4	<i>Principles</i>	11
1.4	GLOSSARY OF TERMS	13
2	HAL	14
2.1	BACKGROUND	14
2.1.1	<i>An Intelligent Environment</i>	14
2.1.2	<i>Equipment and Facilities</i>	15
2.2	HAL'S APPLICATIONS	16
2.2.1	<i>F.I.R.E.</i>	16
2.2.2	<i>Command Post of the Future</i>	17
2.3	HAL AGENTS	18
2.3.1	<i>Metaglu</i> e	18
2.3.2	<i>The Speech System</i>	20
3	THE NATURAL LANGUAGE PROCESSING UNIT	23
3.1	THE GRAMMAR	23
3.1.1	<i>What Is A Regular Grammar?</i>	24
3.1.2	<i>SALLY Uses a Regular Grammar</i>	25
3.1.3	<i>SALLY's Command Grammar</i>	26
3.1.4	<i>Additional Features</i>	27
3.2	THE SCRIPT	27
3.2.1	<i>General Explanation</i>	28
3.2.2	<i>From Input To Output</i>	30
3.2.3	<i>When All Else Fails, Try Xnone</i>	31
3.2.4	<i>Additional Script Features</i>	32
3.2.5	<i>Variable-ize The Output</i>	33
4	AUTOMATIC KNOWLEDGE ACQUISITION	35
4.1	A DYNAMIC DATA STRUCTURE	35
4.2	THREE STAGES: LOAD, LEARN, SAVE.....	36
4.3	WHAT IT LEARNS	37

4.4	WHEN IT LEARNS	39
4.5	HOW IS THE DATA STRUCTURE MODIFIED?	39
5	MULTIMEDIA OUTPUTS	42
5.1	WHY WE WANT MORE THAN SPOKEN UTTERANCES	42
5.2	WHY PLACE THE ACTIONS IN THE SCRIPT	43
5.3	ACTION TAGS	43
5.4	THE ADVANTAGE OF EMBEDDED ACTION TAGS.....	44
5.5	ACTION TAG-AGENT METHOD CORRESPONDENCE.....	45
6	BENEFITS, LIMITATIONS, AND NEXT STEPS	46
6.1	RELATED RESEARCH.....	46
6.1.1	<i>The Intelligent Room</i>	46
6.1.2	<i>Helpdesk Systems</i>	47
6.1.3	<i>Natural Language</i>	48
6.1.4	<i>HCI (Social Agents)</i>	50
6.2	SALLY'S ADVANTAGES.....	51
6.2.1	<i>Advantages For The Programmer</i>	52
6.2.2	<i>Advantages For The User</i>	53
6.2.3	<i>Natural Language Processing Advantages</i>	55
6.3	ISSUES AND LIMITATIONS	56
6.3.1	<i>SALLY's Own Limitations</i>	57
6.3.2	<i>SALLY Inherits Exterior Limitations</i>	59
6.4	EXTENSIONS	61
7	CONTRIBUTIONS	63
	BIBLIOGRAPHY	65

TABLE OF FIGURES

FIGURE 1: HAL.....	15
FIGURE 3: SAMPLE INTERACTION WITH F.I.R.E.	16
FIGURE 4: SAMPLE F.I.R.E. SCREEN DISPLAY	17
FIGURE 5: HAL'S AGENTS.....	19
FIGURE 6: VOCABULARY OF THE SPEECH SYSTEM	21
FIGURE 7: SAMPLE UTTERANCE THROUGHOUT THE SPEECH SYSTEM	22
FIGURE 8: THE NATURAL LANGUAGE PROCESSING UNIT	23
FIGURE 9: SAMPLE REGULAR GRAMMAR	24
FIGURE 10: HYPOTHETICAL GRAMMAR FOR THE LAMP AGENT	25
FIGURE 11: SALLY'S COMMAND GRAMMAR	26
FIGURE 12: EXCERPT FROM A SAMPLE SCRIPT	29
FIGURE 13: SAMPLE USE OF XNONE	31
FIGURE 14: SAMPLE USE OF SYNONYMS.....	32
FIGURE 15: SAMPLE USE OF PRE AND POST	33
FIGURE 16: THE ANSWER USES PART OF THE QUESTION	34
FIGURE 17: LOAD, LEARN AND SAVE STAGES.....	37
FIGURE 18: LEARNING SCENARIO	38
FIGURE 19: MODIFICATION OF DATA STRUCTURE WHEN LEARNING OCCURS	40
FIGURE 20: REASSEMBLY RULES WITH ACTIONTAGS	44

1 A NEW KIND OF HELP

1.1 OVERVIEW

Companies who develop complex systems require helpdesks to support these systems. Although in the past such help has been offered in the form of manuals and technical support on the phone, helpdesk systems have recently moved towards three other types of media: automated telephone systems (“for e-mail help, press 1”), installation CD-ROMs (“Did the test page print successfully?”), and the Internet (e.g. Dell’s “Ask Dudley” [Dell99]).

SALLY is a helpdesk system developed at the MIT Artificial Intelligence Laboratory for use in Hal and The Intelligent Room (two intelligent environments). SALLY’s conception, design and performance constitute the subject matter of this thesis.

SALLY’s goal is to help users interact with Hal, and to help them learn new functionalities in/of Hal. SALLY is based mostly on speech interaction, and its knowledge is a combination of a grammar and a script. SALLY can learn new answers, and therefore develop a more accurate and extended knowledge. In addition, SALLY makes use of multiple kinds of output to aid users; for instance it may act, or display a screen of information, or speak in order to convey its answer.

1.2 CONTRIBUTIONS

The SALLY system achieves numerous contributions, which are reviewed in further detail in **Chapter 7**. Here, I present a brief list to keep in mind. It should provide some perspective on the system, its value and achievements.

The SALLY system:

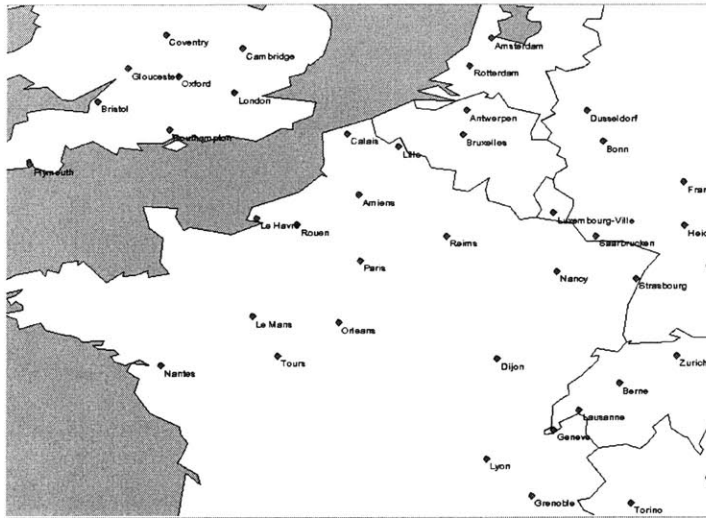
- ❖ Establishes a convenient and useful system that binds together diverse fields of research
- ❖ Demonstrates acquisition of natural, agreeable answers from an automatic helpdesk system
- ❖ Demonstrates that simple language models support useful help
- ❖ Demonstrates incremental and unobtrusive collecting of most helpdesk knowledge
- ❖ Demonstrates ease of multimedia helpdesk responses to user inquiries.

1.3 GOALS AND PRINCIPLES

1.3.1 Scenario

To illustrate what SALLY actually does, I present the following typical interaction.

- SALLY, what can I do? Line 1
- *You can talk to Hal, turn devices on and off, run the command post of the future.* Line 2
- SALLY, what is the command post of the future? Line 3
- *The command post gives geographical information.* Line 4
- SALLY, please show me a map of Paris. Line 5
- *[SALLY displays a map of Paris on the wall.]* Line 6



- SALLY, what else can the command post do? Line 7
- *The command post also understands zoom and move commands. [SALLY zooms in and out to illustrate its answer.]* Line 8



- Thank you. SALLY, how do I change the time on the VCR? Line 9
- *Do you want to set recording times? [SALLY displays the webpage of the VCR's on-line manual, at the Recording a show section.]* Line 10
- SALLY, no, I want to change the clock on the VCR. Line 11
- So, SALLY, how do I change the time on the VCR? Line 12
- *To change the clock on the VCR, press the watch button, then ... [SALLY also displays the Resetting the clock webpage of the VCR manual.]* Line 13

In its last answer, SALLY shows that it learned from previous interactions. It doesn't ask again, "Do you want to set recording times?" as it did in line 10. Instead, it shows that it

learned from the user's previous answer (line 11), and provides an adequate multimedia response for an inquiry regarding the VCR's clock (line 13).

The above scenario presents a simple interaction between SALLY and a user. It is important to note that SALLY understands various questions, that SALLY's answers reveal different layers of detail (e.g. general information, specifics about the command post), that it seamlessly interacts with its testbed (displays the map), and uses various kinds of output (zoom and talk).

1.3.2 Categories of Help

Helpdesks are required to handle a number of topics. In the case of SALLY, these categories are:

- Metaglove (2.3.1) – the distributed agents system SALLY runs on
- The speech system
- The vision system
- The information retrieval systems (2.2.1)
- The embedded physical devices (2.1.2)
- The display systems
- The intelligent multi-modal drawing and sketching program [Weisman98]
- The command post of the future (2.2.2)
- SALLY itself.

1.3.3 Not A Manual

No one will sit down and write a manual about Hal for three reasons at least. No one would read it. It would be obsolete even before it were printed. Finally, it would seem like a complete anachronism given the type of research done in Hal. In addition, it has been observed that at any given time, no single person is aware of the current state of all components of the room. Keeping up with the research done in Hal is a full time job, and one that I want to delegate to the room itself.

1.3.4 Principles

SALLY's development is determined by a set of key principles. These guiding principles are:

- Usefulness,
- User friendliness,
- Robustness,
- Scalability,
- Manual override ability.

1.3.4.1 Usefulness

Indeed, SALLY is not meant as a theoretical stunt, but rather as a pragmatic, useful tool. It is not merely a study in the field of HCI or natural language processing, agent systems or intelligent environments. Rather, SALLY is a system built using the guiding principles previously outlined to solve a real problem: help people effectively.

1.3.4.2 User Friendliness

In addition, SALLY – as a help system - is based on user interaction. I therefore judged it to be fundamental that the user be able to enjoy the interaction as much as possible. This principle of user friendliness carried in its wake several others, among which simplicity (a user should not need help on how to use a help system). Coherence, as well as an intuitive interface (i.e. speech input for SALLY), also followed.

1.3.4.3 Robustness

As in most HCI systems, it is very difficult to predict human behavior. I thus wished the system to robustly handle unpredicted or noisy inputs (the flexibility of the script came in handy).

1.3.4.4 Scalability

Likewise, the domain in which systems such as SALLY will be applied is still somewhat undefined at this point – private or commercial use, small or larger scale. Our scalability principle is meant to ensure that SALLY is not restricted to a certain type of application.

1.3.4.5 Manual Override Ability

Finally, I chose to keep humans in control. At any point in time, a person may override SALLY's answer, modify its script, or shut it down. It is highly desirable to include this ability to manually override a computer system. Indeed, humans generally feel that they cannot understand machines; they feel they cannot predict or control them. As a

consequence, users feel more secure if they know that they can take the upper-hand at any time.

1.4 GLOSSARY OF TERMS

- **AI:** Artificial Intelligence
- **HCI:** Human Computer Interaction
- **Helpdesk:** helpdesk system
- **the Laboratory:** the MIT AI laboratory
- **Metaglu:** computational glue that holds Hal's components together (2.3.1)
- **the Project:** Hal and its related components
- **the Room:** Hal/The Intelligent Room
- **SALLY:** the topic of this thesis; an interactive learning help system

2 HAL

2.1 BACKGROUND

2.1.1 An Intelligent Environment

Hal is an Intelligent Environment [Patch98]. It is a 27' by 37' room containing microphone, computers, projectors, and cameras, and other electronic devices. The room, however, has the appearance of a living room; it contains a door and a window, a TV, a sofa, a coffee table and shelves. Hal provides an agreeable user interface, and allows users to inhabit a smart environment that eases certain tasks. For instance, if a user lies down on the couch, Hal will automatically play some light music and ask the user when she would like to wake up. The possibilities in Hal are endless. In the next section, I present a couple of Hal applications, and the structure of Hal in order to give a brief understanding of the testbed that SALLY runs on.



Figure 1: Hal

2.1.2 Equipment and Facilities

Hal contains the following software and hardware for SALLY to interact with:

- Desktops, laptops,
- Microphones, cameras, TV, VCR, stereo, X-10 devices, projectors,
- The speech equipment (2.3.2): IBM's ViaVoice, British Telecom's Laureate,
- Speech agents that handle regular grammars and the routing of utterances,
- Java and Metaglove (2.3.1)

2.2 HAL'S APPLICATIONS

2.2.1 F.I.R.E.

F.I.R.E. is an application that runs in Hal; it stands for Friendly Information Retrieval Engine. Users designate a particular topic, and F.I.R.E. returns documents that are relevant for this topic. **Figure 3** provides a sample interaction between F.I.R.E. and a user.

```
USER : I need information.
F.I.R.E.: What do you want information on?
USER : Artificial Intelligence
F.I.R.E.: You want information on Artificial Intelligence, correct?
USER : Yes
F.I.R.E.: Please wait while I am looking for the information on Artificial Intelligence ...
        [Then F.I.R.E. presents a screen.]
```

Figure 3: Sample Interaction with F.I.R.E.

Following this dialog, a screen appears that allows the user to refine the search recursively. Once satisfied, the user asks F.I.R.E. for one or more of the documents that it is presented with.

Figure 4 presents the display offered by F.I.R.E. after a user asked for information regarding Travel Agents.

17% of 781 documents have been fetched so far

Potentially relevant phrases:

- **Travel and Vacations Used: Not Yet**
- **Agents and Services Used: Not Yet**
- **Activities and Interests Used: Not Yet**
- **Australia and Oceania Used: Not Yet**
- **Beaches and Islands Used: Not Yet**
- **Countries A-C Used: Not Yet**

1 The most likely categories

- **Cruise Agents** (as in Cruises as in Travel and Vacations)
- **Agents and Services** (as in Australia as in Travel and Vacations)
- **US National Agents** (as in Agents and Services as in Travel and Vacations)
- **Agents and Utilities** (as in Search Tools as in Computers and Internet)
- **Florida** (as in Beaches and Islands as in Travel and Vacations)

- **World**(size: 19) Score: 21.599684 u: false; w: -1
 - **Travel and Vacations**(size: 12) Score: 14.306963 u: false; w: -1
 - ...
 - **Business and Finance**(size: 2) Score: 1.9938973 u: false; w: -1
 - ...
 - **Reference and Education**(size: 2) Score: 1.9938973 u: false; w: -1
 - ...
 - **Computers and Internet**(size: 1) Score: 1.3110318 u: false; w: -1
 - ...
 - **Home, Family and Auto**(size: 2) Score: 0.99694866 u: false; w: -1
 - ...
 - **Society and Politics**(size: 1) Score: 0.99694866 u: false; w: -1
 - ...

Figure 4: Sample F.I.R.E. Screen Display

2.2.2 *Command Post of the Future*

The Command Post of the Future provides geographic and strategic information. The application covers most of the globe – from global maps to very detailed ones. It is also

able to report on meteorological conditions. The Command Post enjoys an interactive interface that handles gestures as well as speech inputs from its users. It understands a variety of commands, such as “show me [a country/a city]”, “zoom in”, and “move north”.

2.3 HAL AGENTS

2.3.1 Metaglu

Distributed software agents run in Hal. The software system that keeps agents happily interacting is called Metaglu [Phillips98]. Metaglu abstracts a layer of details for agent programmers. It is built on top of the Java programming language. Metaglu maintains a catalog of agents running at any given time in the room. It allows agents to rely on other agents, and to specify specific software needs they may have in case they need to be relocated to another platform – for instance vision agents specify that they require an SGI station.

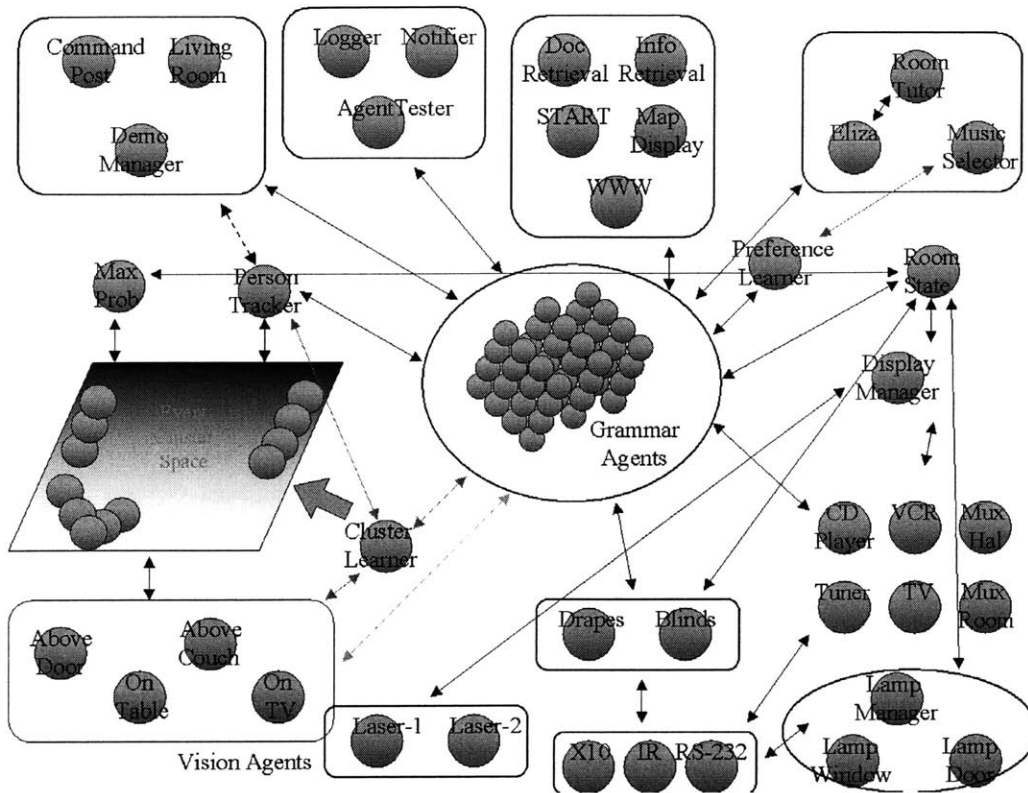


Figure 5: Hal's Agents

Metaglué significantly influenced the design of SALLY – which is why Metaglué is presented here in some details. Indeed, from Metaglué, SALLY gained the idea of small agents, of its interface, and the constraint of its programming language. Mostly, Metaglué made me think of SALLY as an agent rather than a system of its own. If we understand that the behavior of the agent depends not only on its own specifications, but also on those of its environment, then we understand the importance of Metaglué for SALLY – and the importance of being able to seamlessly gather information regarding anything that is going on in The Intelligent Room.

2.3.2 The Speech System

Hal holds an elaborate speech system. At the beginning of the chain comes the speech recognizer (currently IBM's ViaVoice²), which turns sound waves into words. It is closely followed by the speech router which memorizes which agent is interested in which utterance, and sends only interesting utterances to its agents. Each agent also holds its own speech handler which maps utterances to responses. Finally, if the response is vocal, the speech synthesizer (British Telecom's Laureate³) transforms a series of written words into a spoken answer. A microphone and a pair of speakers serve as intermediaries to the speech recognizer and speech synthesizer respectively.

While **Figure 6** clarifies the vocabulary that surrounds the speech system, **Figure 7** presents the different stages of an utterance in SALLY.

² <http://www.ibm.com/speech>

³ <http://innovate.bt.com/showcase/laureate/index.htm>

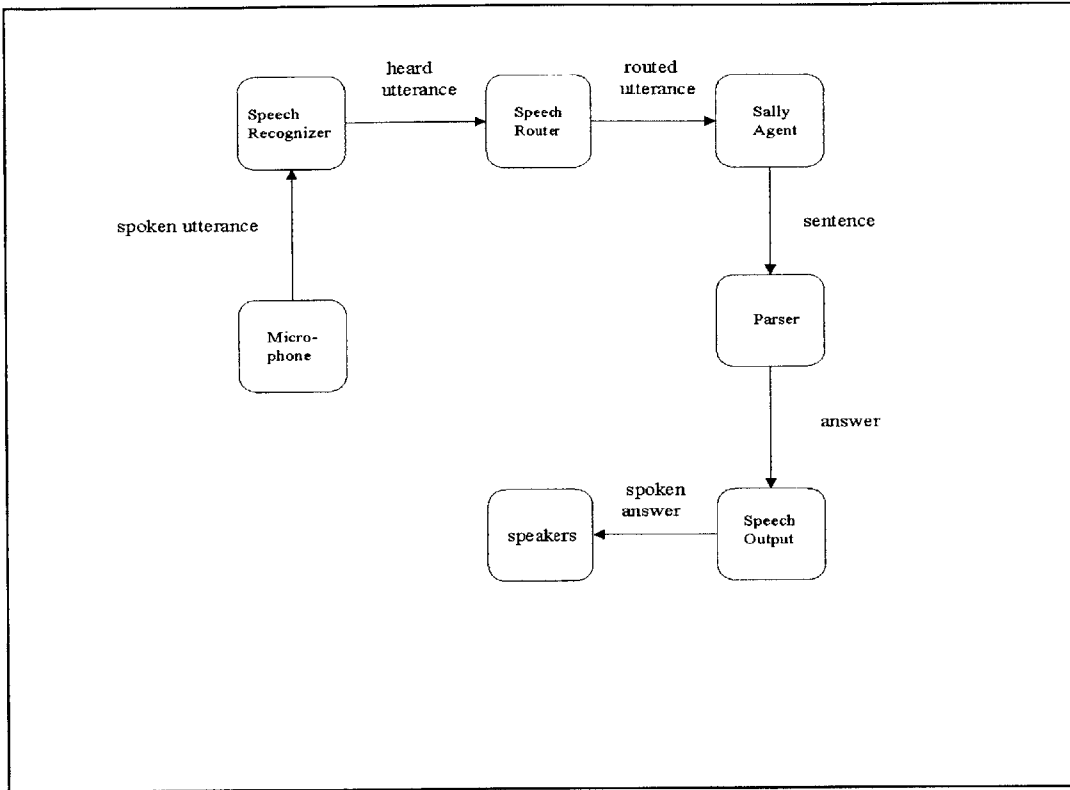


Figure 6: Vocabulary of the Speech System

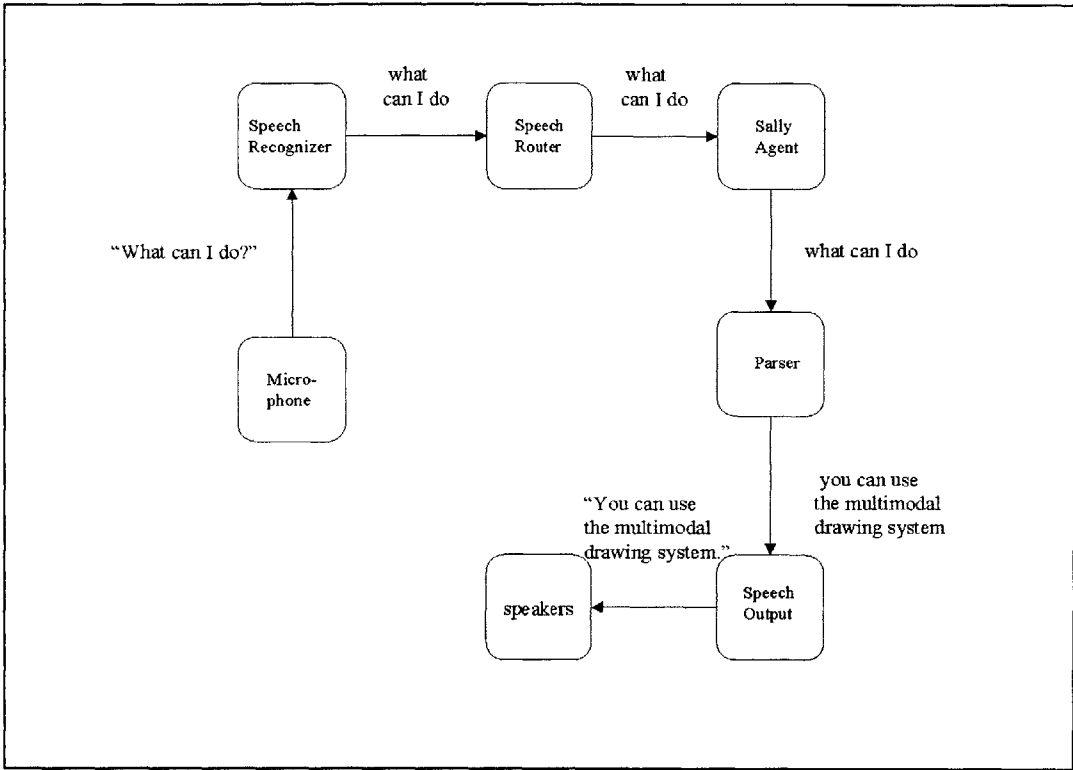


Figure 7: Sample Utterance throughout the Speech System

3 THE NATURAL LANGUAGE PROCESSING UNIT

3.1 THE GRAMMAR

SALLY uses a regular grammar and a script-based system to process natural language. Section 6.2.3.2 provides further discussion on the advantages of combining the two. **Figure 8** shows how the grammar and the script fit together: the command grammar sits on top of the script; both produce outputs.

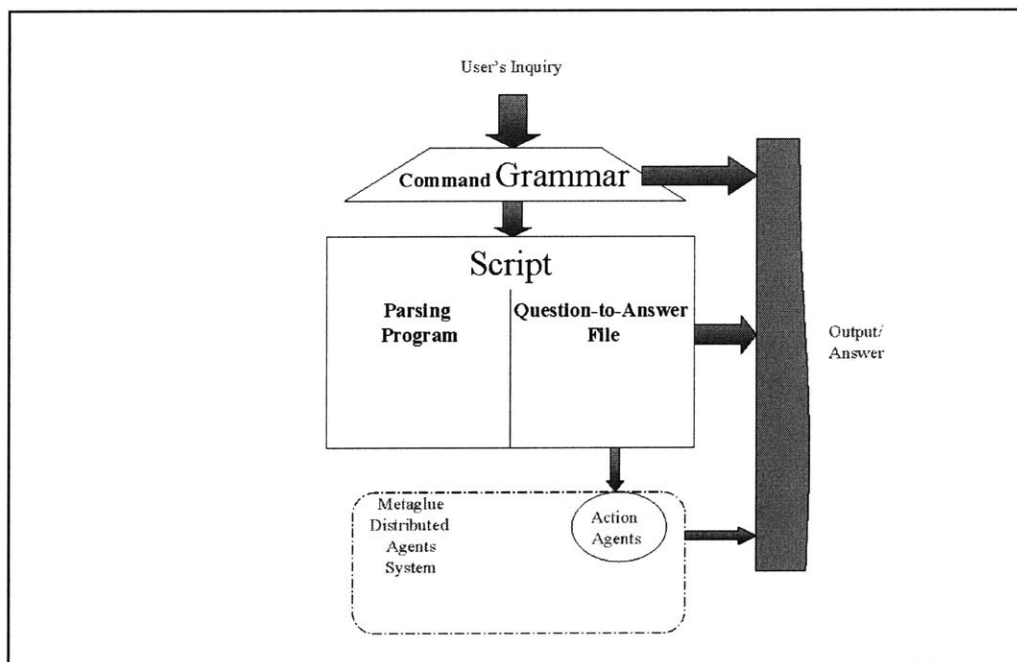


Figure 8: The Natural Language Processing Unit

3.1.1 What Is A Regular Grammar?

SALLY uses a standard Regular Grammar [Gronforst92]. In the case of natural language processing, regular grammars specify how sentences can be structured – from words to phrases to sentences. In other words, grammars describe possible word paths to constitute a sentence – very much like a finite state machine diagram describes possible transitions of a system.

noun = tree forest sequoias
verb = grow exist can be found
adverb = fast densely
adjective = big tall dark somber
article = the a
nounphrase = article adjective noun article noun adjective noun
verbphrase = verb adverb
sentence = nounphrase verbphrase

Figure 9: Sample Regular Grammar

Figure 9 presents a regular grammar. There are two distinct types of lines in the grammar.

- “*article = the | a*” is of the first kind. It defines the category *article* to be either the word “the” or the word “a”.
- “*verbphrase = verb adverb*” is of the second kind. It gives a name to a sub-sentence (in this case, *verbphrase*), and defines it to be exactly – nothing more and nothing less than – a *verb* category followed by an *adverb* category. For instance, “grow

fast”, “exist densely”, and “can be found densely” are all legal verbphrases. “Grow dark”, “exist” or “sequoias grow densely” on the other hand are invalid verbphrases. (*Dark* is not an adverb, *exist* is missing an adverb, *sequoias* is an unwanted noun.)

Therefore the grammar presented in **Figure 9** recognizes sentences such as “Tall sequoias grow fast”.

3.1.2 SALLY Uses a Regular Grammar

SALLY uses a regular grammar to correspond with Hal’s grammar router – any speech-enabled agent in Hal uses grammars. Those grammars let agents register utterances with the grammar router.

```

turnon = action article noun on
action = turn | switch
article = the | a
noun = lamp | it
on = on

```

Figure 10: Hypothetical Grammar for the Lamp Agent

For instance, suppose the lamp agent has registered the grammar presented in **Figure 10**.

Suppose then that a user says, “Turn the light on.”

turnon =	action	+	article	+	noun	+	on
	Turn		the		light		on.

The sentence fits the pattern perfectly. The lamp agent has registered this utterance with the grammar router. Therefore the grammar router can be trusted to relay to the lamp agent that “Turn the light on” is just heard.

In Hal, grammars are kept very simple. Agents often rely on several grammars.

3.1.3 SALLY's Command Grammar

SALLY has a single regular grammar which takes care of sentences such as “SALLY ...”, “Tell me about ...”, and “What do you know about X”. We will refer to this grammar as the *command grammar*. It also manages inquiries such as “Stop”, “SALLY no”, or “Tell me more about that”. Therefore it handles both explicit and implicit user requests.

SALLY's command grammar is presented in **Figure 11**. It is short and simple – yet it is also very effective.

```
grammar hal.allinone;
public <start> = ("I have a question") {question};
public <stop> = ( bye | stop | quit ) {stop};
public <Sally> = sally {Sally};
public <more> = ( tell me more | go on | what else ) {more};
public <no> = (no | nope) {nah};
public <test> = test {test};
public <help> = help {help};
public <thanks> = (thanks | thank you) {thanks};
```

Figure 11: SALLY's Command Grammar

3.1.4 Additional Features

The word “*public*” that precedes all utterances signifies that users may say any of the above – none of the above are for internal purposes only. This makes sense given the interactive nature of SALLY.

The words in curly brackets, such as *{more}* are tags that serve internal purposes. They are sent to a program that handles speech input. This program calls on different methods depending on the tag. For instance, “*more*” means that SALLY is prompted for another answer of the same caliber as the previous one, whereas “*no*” means that SALLY is about to be taught a new answer.

3.2 THE SCRIPT

Despite an apparent contempt in the literature for systems that utilize a simple pattern-matching algorithm in lieu of a complex linguistic model of language – scripts are based on pattern-matching - Weizenbaum, Loner, Hajicova and Reilly present successful systems without linguistic complexity (**Section 6.1.3.2**). In addition, a help system’s knowledge of language need not be profound. A help system is therefore an excellent candidate for a script.

SALLY's script is barely more complicated than a standard one such as Eliza; it only holds certain additional key state variables, and a method to save the script.

3.2.1 General Explanation

A script takes a sentence as input and produces a sentence as output⁴. A script file is a text file that describes how to pattern match on a sentence, and how to produce a sentence as a response to the first one. The pattern matching is done first on keywords, then on patterns including the keyword – i.e. strings of particular characters interspersed by strings of any character in between. For a given pattern that is matched, the script program produces one of several answers.

A script includes the following basic components: *keyword*, *priority number*, *decomposition rule*, and *reassembly rule*. A script file contains several keywords – words of the English vocabulary that are likely to appear in the input sentence. Each keyword is given a priority number. In case several keywords appear in a sentence, the one with the highest priority is given precedence over the others. A list of decomposition rules is also associated with each keyword. A decomposition rule describes a pattern to be matched - it describes how a sentence might be decomposed into fixed words and variable segments. Finally, each decomposition rule owns a list of reassembly rules. Each reassembly rule describes what answer to produce in case the decomposition rule it

⁴ Here we describe scripts that are similar to the one at the core of Weizenbaum's *ELIZA* program [Weizenbaum66].

belongs to is matched. The term *reassembly rule* comes from the fact that each reassembly rule describes how bits of sentences are reassembled into an answer.

Therefore the script is a series of keywords, priorities, decomposition rules and reassembly rules. An excerpt from a sample script might look like:

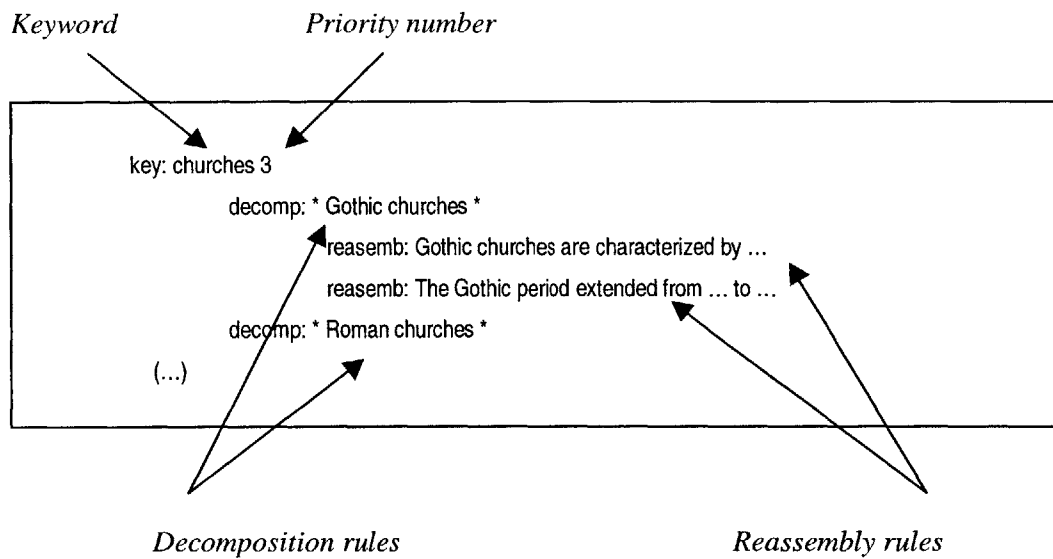


Figure 12: Excerpt from a Sample Script

Figure 12 specifies the following script structure:

- | | |
|-----------------------|---------------------------|
| ◆ Churches | keyword |
| ◆ 3 | priority number |
| ◆ * Gothic churches * | decomposition rule |

◆ Gothic churches are characterized by...

reassembly rule

One final word on decomposition rules: a star (*) means, “*match anything*”. “* *Gothic churches* *” is the actual pattern of words to be matched – it is not the name of a rule, but the rule itself. And the rule specifies that all of the following

<i>What do you know about</i>	<i>Gothic churches</i>	?
<i>Tell me more about</i>	<i>Gothic churches</i>	.
<i>When were</i>	<i>Gothic churches</i>	<i>built?</i>

match the pattern specified in

* *Gothic churches* *

3.2.2 From Input To Output

The script is associated with a program that builds a key stack of all keywords listed in the script – ordered from highest to lowest priority. Starting with the first one on the stack: if keyword *k* appears in the input sentence *S*, then the script program attempts to match *S* with the first decomposition rule *d* associated with *k*. If this decomposition rule fails, the program tries to match *S* with *d+1* – *k*'s next decomposition rule. If all of *k*'s decomposition rules fail, the program tries to match *S* with the decomposition rules of *k+1* – the next keyword on the stack that appears in *S*.

When decomposition rule *d* is matched for the first time, the script's answer is the first reassembly rule *r* associated with *d*. If *d* is matched again, the answer is *r+1* – *d*'s next reassembly rule. In other words, the script program rotates through the reassembly rules of a decomposition pattern.

3.2.3 When All Else Fails, Try Xnone

The script program calls on the “*xnone*” keyword when none of the keywords appear in the input sentence, or when all decomposition rules fail to match. *Xnone* is a script's default match. For instance, a script might handle it as follows:

```
key: xnone 0
  decomp: *
    reasemb: I didn't understand, could you repeat please?
    reasemb: Can you phrase that differently perhaps?
    reasemb: Would you like to learn more about the command post of the future?
    reasemb: Do you want to learn about Luke's drawing system?
    reasemb: I'm having trouble understanding you.
    reasemb: Please speak more clearly.
    reasemb: Say that again?
    reasemb: Can you repeat please?
```

Figure 13: Sample Use of Xnone

The priority of 0 parallels the fact that *xnone* is a default – therefore it should not take precedence over any other possible match.

3.2.4 Additional Script Features

3.2.4.1 Synonyms

On top of the basics come some convenient apparatus to handle sentences in an intelligent and sophisticated manner. One of them is the concept of synonyms. The scriptwriter may declare synonyms at the beginning of the script. This allows that the scriptwriter does not have to duplicate decomposition and reassembly rules for every synonym. The use of synonyms in a script looks like:

```
synonym: C sharp | D flat  
synonym: C | do  
synonym: E | mi  
synonym: A | la
```

Figure 14: Sample Use of Synonyms

3.2.4.2 Pre and Post

Pre and *Post* are two other of these useful script features. *Pre* handles the conversion of sentence bits before the sentence goes through pattern matching. *Post* handles the conversion after the pattern matching. For instance, “my” in a question gets translated to “your” in the answer.


```
pre: certainly yes
pre: nope no
post: my your
post: mine yours
post: am are
```

Figure 15: Sample Use of Pre and Post

3.2.5 Variable-ize The Output

Yet another useful characteristic of scripts is the possibility to reuse parts of the question in the answer. This allows us to set the output as a variable that depends on the input. For instance in **Figure 16**, (2) corresponds to the second star in the pattern. Therefore if a user asks “Do you remember Bastille Day 1989?”, the script program will answer “Did you think I would forget *Bastille Day 1989?*”. If, however, a user asks “Do you remember my birthday?”, the script program will answer “Did you think I would forget *your birthday?*”.

```
keyword: remember 5
  decomp: * do you remember *
    reasemb: Did you think I would forget (2)?
  ...
```

Figure 16: The Answer Uses Part of the Question

A script is the association of a text and a parsing program. Together they take a sentence and produce a (corresponding) response.

4 AUTOMATIC KNOWLEDGE ACQUISITION

Here is a simple example of how automatic script modification occurs. Say an inexperienced user walks into Hal with one of the people working on the project, and asks SALLY to tell him how to *magnify the current display*. SALLY currently does not have in its script or grammars any knowledge of how to handle this question. It might answer that it does not know. At this point, the person who works on the project can simply say, “SALLY, no, to magnify say Zoom in”. SALLY will parse the sentence, recognize that this is an attempt to correct its script, and add a new decomposition pattern and reassembly rule to its internal data structures. If the inexperienced user tries again “SALLY, how do I magnify the display?”, SALLY will this time answer “To magnify say Zoom in”.

4.1 A DYNAMIC DATA STRUCTURE

Scripts are typically understood to be clumsy, outdated, and static systems. The SALLY system, however, is able to modify its knowledge over time. This dynamic characteristic is at the core of the system; it allows the system to learn additional knowledge. It is also one of the most powerful features of SALLY.

One of the early concerns was that I might not be able to dynamically change the internal data structure that holds the knowledge of the system. If this were the case, I would have had to reload a script every time any change had to be made to the knowledge data structure. This concern turned out to be unfounded.

4.2 THREE STAGES: LOAD, LEARN, SAVE

There are three stages that allow SALLY to have a dynamic data structure – load, learn and save. From a static script file, SALLY builds a data structure that contains the equivalent information (*load* stage). During interactions with users, SALLY accesses this data structure and modifies it (*learn* stage). At the end of an interaction, it saves the new data structure back in script file format (*save* stage). Therefore, when SALLY runs next, the newly learned knowledge is loaded. Learning is not lost.

The following diagram summarizes adequately the state of the system during load, learn, and save modes.

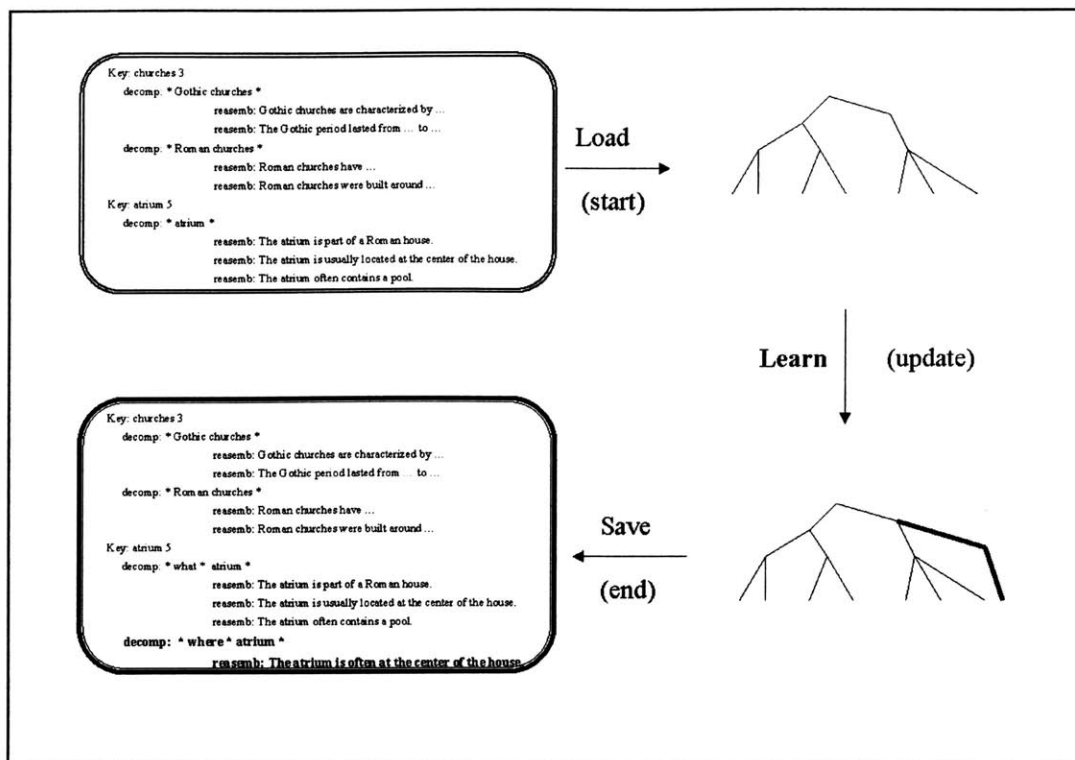


Figure 17: Load, Learn and Save Stages

4.3 WHAT IT LEARNS

What SALLY learns from its users are pairs of a question and an answer. Indeed, it learns to optimize its answers and make them as helpful as possible.

Let us use the **(q, a)** shorthand to signify a question and its answer. If at one point SALLY's knowledge contains the **(q, a)** pair, and that the answer **a** is judged to be

inadequate, then SALLY will learn a better **(q, a)** that will have precedence over the old **(q, a)** pair.

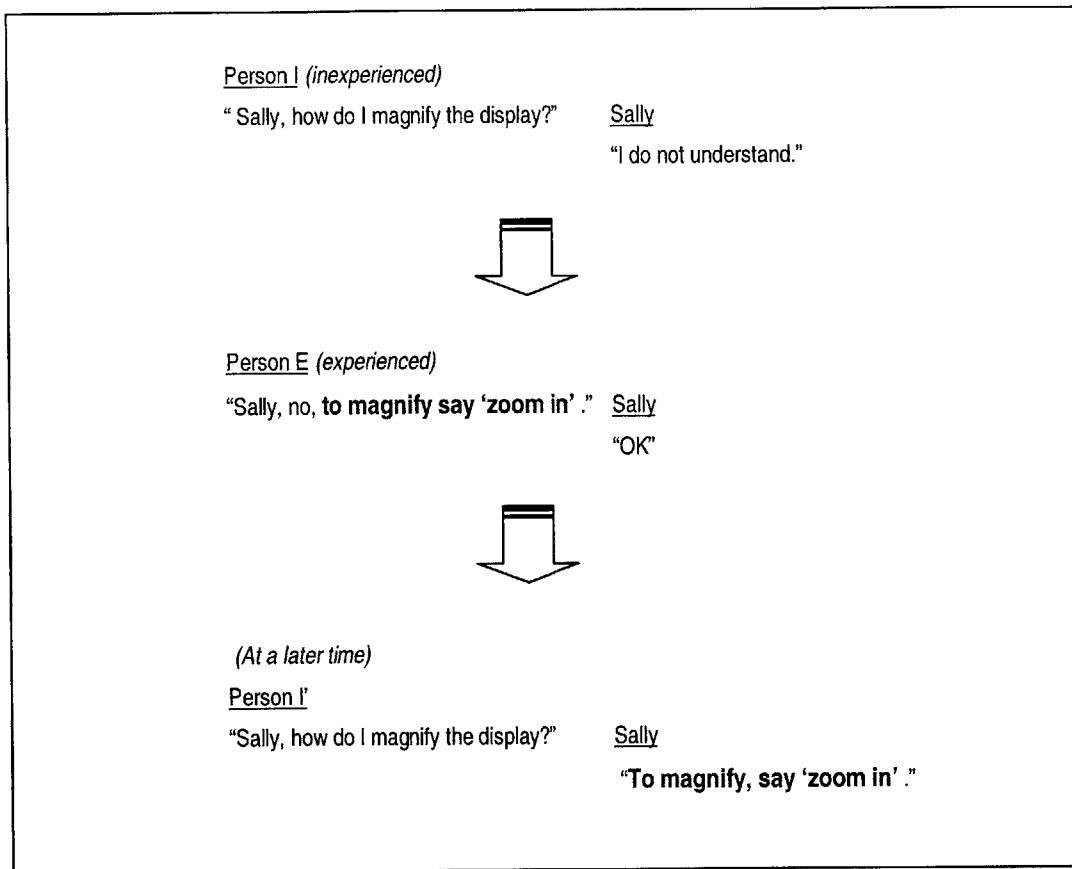


Figure 18: Learning Scenario

4.4 WHEN IT LEARNS

SALLY acquires knowledge when it is told that its answer is unsatisfactory. Starting a sentence with “SALLY, no” signals that the previous answer was inadequate, and that someone is about to provide a better answer that SALLY should remember.

In other words, SALLY learns when three conditions are met:

- ❖ Someone asks for help,
- ❖ SALLY’s answer is insufficient,
- ❖ Someone provides a better answer.

4.5 HOW IS THE DATA STRUCTURE MODIFIED?

SALLY does not exactly hold a table of questions and answers. The (question, answer) pair metaphor corresponds to the input provided to the script program and the output produced by the script program. To learn a new answer means to modify the structure of the script. What does SALLY really do to its internal data structure when it is told that its answer is disappointing?

Question **q** triggers answer **a**. Question **q** is matched in the script with keyword **k**, priority **p**, decomposition rule **d**, and reassembly rule **a** (the answer). Then SALLY is taught a better **a'** – for the exact same **q**.

Thus the sub-tree hanging from the current keyword **k** is modified: a decomposition rule **d'** is added that matches **q** exactly. To this decomposition rule I provide a reassembly rule: the desired answer **a'**. (This holds true even if keyword **k** is the default xnone keyword.) **Figure 19** presents a sample modification of the data structure.

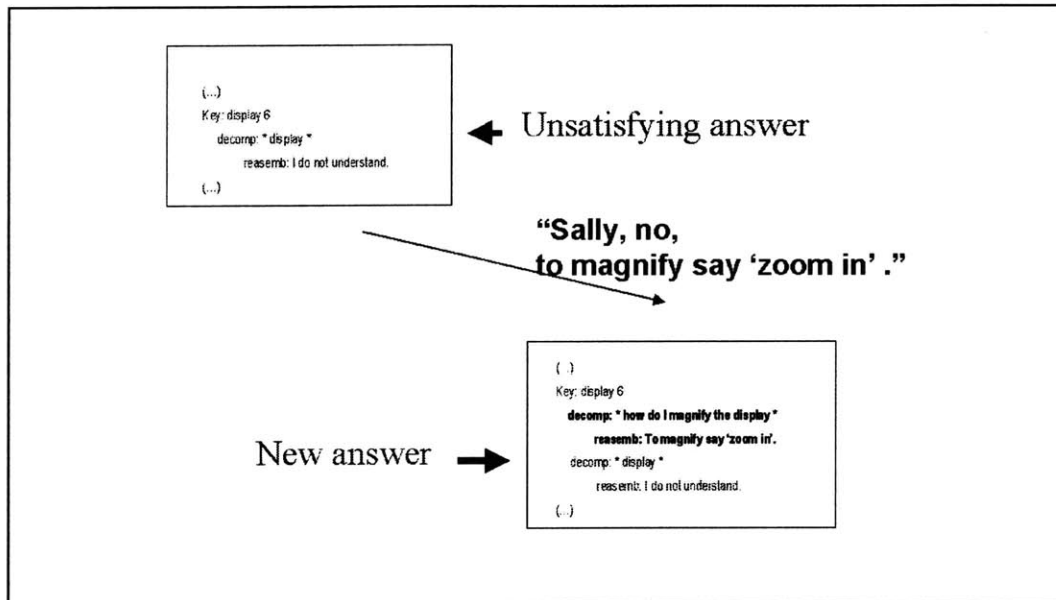


Figure 19: Modification of Data Structure when Learning Occurs

The logic is that if sentence **q** triggers keyword **k**, then sentence **q** again will still trigger **k** in following interactions. It is guaranteed that next time **q** is spoken, keyword **k** will be

accessed and decomposition rule **d'** will be matched. In other words, from then on, **a'** is assured to be the answer any time **q** is the question.

As yet, I do not modify either the priority or the keyword. If I did, I would lose the guarantee that the script program behaves as expected. For instance, certain questions, which are currently correctly answered, might then match another decomposition and produce an incorrect answer. Or the current question **q** might match another decomposition rule in another keyword – the learning would be wasted.

It is conceivable, however, to teach SALLY new keywords and rules. This interaction would require a slightly more complicated dialog design. It would be useful, nonetheless, if a new category were to be added to the system supported – such as adding a new application in Hal.

5 MULTIMEDIA OUTPUTS

Along with automatic script acquisition, one of SALLY's most salient features is its ability to juggle outputs of different kinds. Not only does it use vocal feedback, but it also displays information in order to keep a low-intensity vocal channel, and presents some sample actions and interactions at the same time as they are explained. Therefore the help system is not limited to one media.

5.1 WHY WE WANT MORE THAN SPOKEN UTTERANCES

The script as described above is a standard script and one that is used as a start in SALLY. Very soon, however, it became clear that I wanted more than just spoken utterances as responses to our questions. This came about for four reasons. Not only would it make the system much more *fun and interesting*. It would also make SALLY *a better pedagogical tool* - varying the kinds of output and calling on several senses of the user increases the user's chances of learning and remembering. In addition, this would allow us to *shorten the vocal responses* if it combined them with other feedback. Finally, The Intelligent Room and Hal are highly dynamic and multi-dimensional environments that are *best described with a combination of feedback methods*.

5.2 WHY PLACE THE ACTIONS IN THE SCRIPT

I have included actions in SALLY by embedding them in the script – together with the regular string of text. Our reasoning is the following. The script’s output is examined in any case - and spoken out loud in the old case. Therefore it does not add much work to look for action tags at this point. The other reason is that adding the actions anywhere else would require a table specifying which actions correspond to which sentences, whereas no such table is needed if actions are directly included in the script. By embedding actions in the script, parsing and matching efforts are not duplicated.

5.3 ACTION TAGS

Action tags look like “*!zoomin*”. They are one-word tags preceded by an exclamation point. Sentences in the script no longer look like every day sentences, instead every token in the sentence starts with a tilde, “~”. Let’s assume that I want to say, “Let’s start the music”, and accompany this utterance with music starting to play. Then a reassembly rule would look like:

```
reasemb: ~Let's start the music ~!startmusic
or
reasemb: ~Let's ~!startmusic ~start the music.
```

Figure 20: Reassembly Rules With Action Tags

The second sentence in **Figure 20** reveals that it is possible to embed actions in the middle of vocal responses.

5.4 THE ADVANTAGE OF EMBEDDED ACTION TAGS

It is highly beneficial that the location of action tags is not restricted to the end of a sentence: it adds much flexibility to the SALLY system. There is no need to wait until the end of a sentence to trigger an action. In addition, in case a sentence is to trigger several actions, those actions do not have to be round up at the end of a sentence. Instead, they may be scattered throughout the sentence as is appropriate. Besides, in some cases, embedded actions are the only action tags that make sense – particularly if I were to describe two opposite actions in the same sentence. In this last case, I can describe the first action, trigger it, then describe the second one and trigger it. For instance, the sentence “You can say ‘Zoom in’ or ‘Zoom out’.” naturally becomes

“~You can say ‘Zoom in’ ~!zoomin ~or ‘Zoom out’ ~!zoomout.”

5.5 ACTION TAG-AGENT METHOD CORRESPONDENCE

With the action tags described above, the system still needs a way to decrypt what to do with those tags. As explained above, and for reasons of efficiency, this parsing is done right after the script program returns its answer to the SALLY Agent. As the system stands now, the agent looks for action tags it knows. For each known tag, it calls on a method of the agent.

For instance, say we go back to the *!zoomin* example, then the SALLY Agent calls the Zoom method of the Display agent with adequate parameters.

6 BENEFITS, LIMITATIONS, AND NEXT STEPS

6.1 RELATED RESEARCH

This project touches upon many areas of research, and each of these areas encompasses a vast literature of its own. I restrict this section to the references that directly inspired me. I will first cover The Intelligent Room and related projects. A brief overview of research on helpdesk systems follows. Finally, I will look into natural language issues, and the field of social agents.

6.1.1 *The Intelligent Room*

6.1.1.1 Research Done in The Intelligent Room

The Intelligent Room is the testbed on top of which SALLY was developed. In order to get a better understanding of the various components of the room, one might want to look at Coen's AAAI paper on Principles for Building the Room [Coen97], and the MIT Artificial Intelligence abstract on the project [Coen98]. [Coen99] describes the importance of Context-Enhanced Speech Recognition.

6.1.1.2 Organization of Agents

All components in the room are represented as simple software agents organized in societies – a concept inspired from Minsky's *The Society of Mind* [Minsky88]. Agents run on Metaglua, an extension to the Java programming language that provides high-level

support for writing large groups of distributed software agents that interact with each other [Phillips98].

In addition, the agents are organized in layers – where each layer specifies a new, more complex level of behavior. This model is inspired from Brooks' model of Subsumption Architecture [Brooks85].

6.1.1.3 Other Intelligent Environments

The Intelligent Room is not the only room of its kind. The following projects also investigate intelligent environments:

- Reactive Room, Toronto [Cooperstock97]
- KidsRoom, MIT Media Lab [Bobick98]
- Visualization Space, IBM [Lucente98]
- Neural Network House [Mozer98]
- Classroom 2000 [Abowd96]

To get a more thorough account of the current research in Intelligent Environments, one should look into the AAAI Intelligent Environments Symposium, 1998 [IE98].

6.1.2 Helpdesk Systems

Many companies have invested large funds and efforts into implementing helpdesk systems for their clients. Currently, most helpdesks are staffed by humans who answer

telephone calls. As part of a growing trend, certain companies now possess automated telephone answering systems (“*For this, press that*”), or intelligent telephone systems (“*What stocks, and how many, would you like to buy?*”). Examples include SUNDIAL – an intelligent transportation system, Verbmobil, Bell’s Automated Alternate Billing Services, and SPEECHtel [Bernsen98]. Yet other helpdesk systems keep the intelligent part and get rid of the speech interaction part: they place helpdesk software agents on the Internet [Dell99].

Nuance Communications is one of few companies developing intelligent telephony systems [Nuance99].

6.1.3 Natural Language

Most of the sophisticated helpdesk systems mentioned above employ highly complex speech models. Our approach is radically different.

6.1.3.1 Eliza

Weizenbaum created a program called Eliza [Weizenbaum66] – a script-based system that responds to typed user input with a typed response of its own. Eliza simulates the characteristics of Rogerian psychologists. Later, various tutorials were developed from the Eliza system, including tutorials for n-Th dimensional physics [Hayward68]. [Eliza68] explains the nuts and bolts of writing scripts for the original Eliza system.

Charles Hayden coded Eliza in Java [Hayden98]; his code was used as a starting point for integrating Eliza in Hal.

6.1.3.2 Scripts As A Representation Of Language

One might wonder why scripts were used in SALLY. In her *Linguistic Aspects of Natural Language Processing*, Hajicova writes, “It has been noticed that even systems modeling certain kinds of dialogues may be formulated without a detailed complex linguistic analysis (cf. Weizenbaum’s ELIZA).” [Hajicova92, p.11]

Lenny Foner has studied *Julia*, a chatroom bot. He notices, “one of the interesting things about Julia’s construction is that her parser is shockingly simple, as such things go. It is barely more complicated than Eliza’s parser in some ways, in that it does *not* decompose its inputs into parse trees or anything else that a linguist might suggest. Instead, it does a very simple pattern-match on its input, looking for particular strings of characters separated by strings of any characters in between.” [Foner97]

Reilly at CMU developed a simulation of baseball card trading. His characters also handle natural speech as a “search for patterns of keywords for language understanding”. [Reilly96, p.115] Whalen describes another such system, CHAT [Whalen96].

Finally, many others have investigated what is the right model for natural language. Winograd designed a procedural model for natural language understanding [Schank73].

Katz considered the lexical properties of language, and how to exploit them [Katz88]. Zue developed a natural language speech recognition system [Zue92]. And Horacek introduced a new model for integrating user inferences when modeling user behavior [Horacek97].

6.1.4 HCI (Social Agents)

The “*Eliza effect*” refers to humans’ willingness to believe computers - our tendency to anthropomorphize computers and interfaces [Don92]. Reeve and Nass further investigated this tendency of humans to treat computers and media as real people and places [Reeves96]. Also in Nass, Steuer, Tauber [Nass94].

Tannen produced famous sociology research on people interaction, and the importance of word choice [Tannen97]. Word choice also has ramifications for the purposes of word choice in human-computer interactions. In addition, the gender of a computer’s voice is important: research reveals that people are less likely to respect a female voice and believe its technical knowledge [Reeves96].

From research such as the Oz project at CMU [Bates92], researchers have discovered some key concepts for designing conversational interactive agents. Particularly, feedback and succinct answers are fundamental characteristics of a successful social agent. More on agent personality in [Ibister98, Foner97, Picard97, Rousseau97, Walker97, Reilly96, Bates94, Laurel90].

MIT Professor Murray uses Eliza script to let her students create fictional characters in a class on Interactive Fiction [Murray97, Murray98, Murray99]. She explains that restricting or stereotyping an agent's personality effectively constrains the field of interaction between the human and the synthetic agent. As a consequence, conversations between humans and such agents are more likely to be successful – because humans know what realm to interact in. In a similar manner, Reilly constrained his world to baseball card trading in order to demonstrate effective communication between his fictional characters [Reilly96].

6.2 SALLY'S ADVANTAGES

The SALLY System offers many advantages over other designs, and other Help systems. Indeed, SALLY's approach shows to be highly beneficial not only from a programmer or user's point of view, but also from a natural language processing perspective.

6.2.1 Advantages For The Programmer

6.2.1.1 Minimal Initial Information

For instance, there is minimal initial information to be entered by a programmer⁵. A minimal command grammar and a small script suffice. In SALLY, the grammar is on the order of two dozen lines, and the initial script is around 50 lines long. This represents an incredible benefit compared with other systems that require all information to be manually entered. Instead, from its compact initial information, SALLY is able to learn all further necessary knowledge. There is no limit on the length of the final script nor on the amount that SALLY can learn.

It is also important to note that the quality of SALLY's learning does not depend on the initial information; therefore even a flawed initial script can evolve into a quality component of the help system.

6.2.1.2 Small Agents

Finally, I have chosen to remain faithful to concepts from *The Society of Mind* [Minsky88] and the Subsumption architecture [Brooks85]. The project is thus composed of small units (agents). This type of organization of software agents makes it much easier for a programmer to customize the project.

⁵ By "programmer" we mean the person in charge of setting up the knowledge of the help system.

6.2.1.3 Simple Natural Language Processing

SALLY is extremely simple; it contains no model of natural language, except for a simple command grammar that specifies a couple of sentence structures⁶. Some might view it as a weakness. In our case, it is in fact a strength that makes SALLY a more flexible system. It is almost trivially easy for SALLY to learn new utterances. In addition, this learning is not complicated by trying to fit new utterances into a given language model.

6.2.1.4 Control Over New Knowledge

Humans teach SALLY new knowledge. Therefore the content of SALLY's knowledge is known at all times. This is invaluable compared to other systems that attempt to correlate events, or try to pick up new answers self-handedly. Those sometimes result in very unexpected answers, such as a system correlating two seemingly independent parameters. The control over new knowledge is also extremely important for a Help system. How might a user react if SALLY answered some nonsensical or random sentence?

6.2.2 Advantages For The User

6.2.2.1 Easily Multilingual

The simplicity of the natural language model brings another surprising yet interesting consequence: SALLY is easily multilingual.

⁶ Most speech recognizers, however, do embed some internal model of language. IBM's *ViaVoice* is one of them.

In order to have SALLY in, say, French, all it needs is a French speech recognizer and a French speech synthesizer, a French command grammar and a French script. The first two can be found off the shelves. The French command grammar would still be around two dozen lines long. And the French script could be the exact translation of the English one.

6.2.2.2 Multiple Users

SALLY also reveals flexibility in another dimension: it handles multiple users as well as a single user – as long as all are wearing microphones.

6.2.2.3 Natural Utterances

Finally, SALLY's natural utterances are very pleasing. They are not forced into a proper model of speech, and they are less likely to be as awkward as they might be if they had been learned some other way. Indeed, sentences that SALLY learns are sentences that are spoken by humans. Hopefully they will not sound as affected as some current help systems do.

6.2.2.4 Natural Interface

SALLY's interface is one of spoken natural language – which is most intuitive. SALLY's feedback is also very unobtrusive (i.e. mostly chirps; SALLY also automatically detects the end of the sentence.) In addition, users can say "*Please tell me more about this*", or "*Go on*"; and SALLY will provide additional information about the previously accessed topic – there is no need to remind SALLY of the topic.

6.2.3 Natural Language Processing Advantages

6.2.3.1 Automatic Knowledge Update and Multimedia Output

The automatic knowledge acquisition (**Section 4**) and the multimedia outputs (**Section 5**) are two essential features of SALLY. They are a step ahead of most existing Help systems. They allow to be entirely dynamic. They provide endless opportunities for upgrades, refines and add-ons. They allow users to make diverse as they come to mind, and on the fly. Finally, they allow SALLY to vary in its output and present its answer in a most convincing and interesting manner.

6.2.3.2 Script-Grammar Combination

SALLY handles natural language processing by means of a grammar and a script-based system. The alliance of a script and a grammar opens up the natural language possibilities enormously, and produces the core of SALLY's power.

In SALLY, the *script* allows us to hold knowledge easily, and learn easily. The script is flexible and modifiable – therefore SALLY's knowledge is dynamic - as opposed to static. In a script, it is easy to ask “*match anything here, then tell me what you matched*”. In addition, the script holds context information by remembering which was the last question asked and what was the last answer given. Finally, the script serves as the memory of the system. When a user says, “*Tell me more about that*”, the natural language processing unit is able to make sense of the word “*that*”; it answers accordingly.

The *command grammar*, on the other hand, is used as a smart door or filter to the script. Thanks to it, only sensible utterances are passed onto the script. This is very important for our purposes. If it were not the case, the script could be asked to handle an abnormal utterance – and the user certainly wouldn't understand the script's answer. Besides, without the grammar, the script would not be able to tell the difference between regular interaction with Hal and a question for SALLY. A command grammar can also call on other pieces of code – including the script itself – when it returns information. Therefore from within the natural language processing unit, I am able to call on other parts of the unit, even exit temporarily and come back to it without losing context. Finally, the command grammar roots SALLY in Metaglove and the world of speech agents.

Together, the script and the command grammar form a coherent unit that exhibits robustness and flexibility.

6.3 ISSUES AND LIMITATIONS

The SALLY system includes a number of limitations. In this section I present them and explain why they might be acceptable.

6.3.1 SALLY's Own Limitations

6.3.1.1 The Learning Depends on Most Words

SALLY's learning scheme depends on almost every word. When it learns the answer to a question, the system does not necessarily learn the answer to a close variant of the question. This limitation, however, is eased by the fact that scripts handle synonyms. It is therefore possible to define synonyms that handle many cases where users might formulate the same question differently. For instance, "*What is X?*" can be specified as a synonym of "*What do you know about X?*"

6.3.1.2 SALLY Holds No Pragmatic Knowledge

SALLY does not currently hold any pragmatic knowledge, and in particular it has no knowledge about pronoun disambiguation; for example it cannot always deduce that the word "it" means "the door" in the current context. There are two reasons behind this. First, the system does not hold an explicit knowledge of the state of the room around it. Secondly, pronoun disambiguation is a problem that has yet to be solved. The second reason sheds a new light on the first one, and explains - or perhaps forgives - our choice to allow the room to have only an implicit understanding of its state.

6.3.1.3 SALLY Does Not Learn By Itself

SALLY gets taught; SALLY does not grasp new knowledge by itself. This certainly necessitates that users are proactive and willing to make SALLY improve. At the same time, teaching SALLY allows users to be in control of SALLY's knowledge. SALLY does

not learn coincidences or noise - like other learning schemes might. Instead, it learns human utterances in a well-defined context.

6.3.1.4 SALLY Does Not Learn New Actions

SALLY's script holds speech and action components. However, SALLY only learns speech - no actions. This comes from the fact that behind every action is a piece of code that would have to be written ahead of time (and therefore the action tag could be entered manually). Users would have to say something along the lines of "SALLY, no, speech to magnify the display say zoom in, action z-o-o-m-i-n". Users would then specify the lines of code that correspond to the *zoomin* action. I decided against such awkwardness, although one could extend SALLY with an additional interface in order to deal with learning action code.

6.3.1.5 Compound Phrases and "Aren't you"'s

Scripts typically handle compound phrases and "aren't you"'s poorly [Shieber93]. SALLY does have a rule to respond to "aren't you" 's adequately (i.e. to ignore them instead of turning them into "am I not "'s). Compound phrases remain an issue, however. It is unlikely, though, that they will be used frequently enough to disrupt a conversation between SALLY and a user.

6.3.2 SALLY Inherits Exterior Limitations

To introduce spoken interactions in the system is a double-edged sword. On the one hand, it greatly facilitates the interaction between the help system and the user. The user does not have to stop everything and go over to a terminal to get help. On the other hand, it introduces an inherent dependency on the quality of speech recognition and speech output software. Both speech recognition and speech synthesis have significantly improved in the last couple of years. However, such dependency remains a concern.

6.3.2.1 Speech Recognition Limitations

In terms of speech recognition, the pattern matching algorithm, the grammars, and the script all depend on the sentence that is the output of the speech recognition system. Therefore they all depend on the accuracy of the speech recognizer. If “turn on the lamp by the door” is understood to be “turnip the lamp by outdoor”, there is very little chance that the system will answer adequately to the user’s real question. We are currently working on ways to correlate several agents’ knowledge, but SALLY is limited to the sentence that the speech recognition software passes along, however wrong it may be⁷.

Furthermore, the *script modifications* are also dependent on the quality of the speech recognition software. As the system stands now, someone says, “SALLY, no”, then provides a better answer. If this answer is misunderstood, it is consequently incorrectly

⁷ One of the current workarounds of the IBM system is to bias the speech recognizer for certain utterances. This is, however, clearly insufficient and not an acceptable alternative to an improved speech recognizer.

entered in the script. The next time that any user prompts SALLY for that particular answer, the user is likely to be perplexed.

6.3.2.2 Speech Synthesis Limitations

The issues related to speech synthesis are slightly different. They are more subtle and psychological. We have come a long way from early synthesis systems that sounded closer to bangs on a tin can than to any human voice. At the same time though, work remains to be done - especially in the areas of intonation, of sentence melody and dynamics. It is especially annoying to hear the same sentence uttered in the exact same way 15 times a day. Therefore I have explored other, less obtrusive means of providing feedback to the user (such as chirps and non-vocal feedbacks), and I have attempted to keep the utterances spoken by SALLY short and relevant.

Most of SALLY's limitations are not inherent; they are merely the result of its interaction with other components of Hal.








6.4 EXTENSIONS

This table presents wishful extensions to the SALLY system, and how they might be implemented.

What To Do Next		How To Get It
Maintain a compact script		Include one question a day from SALLY to users about possible script changes
Correct misheard utterances		Include one question per greeting
Delete obsolete utterances		Clean up the script manually
Combine similar utterances		Combine sentences that are >90% similar
Adjust the responses to the level of the user		Include a user model
		Parameterize responses: use frames [Winston78]; modify the reassembly rules in technicality depending on the frame slots
Depend less on the speech recognizer		Combine several speech recognizers
		Use other room knowledge
Depend less on the syntax of the decomposition rules	Regular expressions to relax the decomposition pattern	

(What I Want Next)

(How To Get It)

Not have to wear a microphone		Embed microphones in the room in places where users are likely to speak
Not have to call on SALLY		Utilize eye gaze information to detect when a user is speaking to SALLY
Provide better (entertaining?) default responses		Implement an ACM trivia agent
Be able to playback interactions		Expand SALLY's memory: keep track of the last X utterances, or last X minutes of interaction
Include more context information		Include self-watch and reflect capabilities in Hal
		Expand the natural language processing capabilities of SALLY
Learn actions		Include additional code for interaction with user to learn

7 CONTRIBUTIONS

In this section, I review the contributions made by this thesis – as they have been pointed out in **Section 1.2**.

First, the SALLY system successfully brings together research from intelligent environments, helpdesk systems, natural language, and social agents in human-computer interaction (**Section 6.1**), and generates a novel, convenient and useful helpdesk system.

Second, the SALLY system offers natural answers as a gratuity. Indeed, most answers are utterances that SALLY learned from users (**Section 4.3**). From such naturally acquired sentences, SALLY makes the point that it is possible to *obtain natural, agreeable answers from an automatic helpdesk system* without a full-fledged dialog and interaction a priori-design.

SALLY's natural language processing unit is uncomplicated. It is the combination of a parsing script and a regular grammar (**Section 3**). Yet it is effective and provides appreciated help to users. Therefore SALLY shows that *simple language models can support useful help*.

In addition, the help system described in this thesis starts off with minimal initial information. SALLY successfully compiles new knowledge – while it is running (**Section**

4). In other words, *most help knowledge can be collected incrementally and unobtrusively.*

Finally, SALLY provides various kinds of answers: spoken utterances, displayed information, actions (**Section 5**). This diversity of outputs is beneficial for the user's learning. It is also less obtrusive than mono-media help. SALLY successfully produces multimedia responses to questions and demonstrates *the ease of multimedia helpdesk responses to user inquiries.*

BIBLIOGRAPHY

- [Abowd96] **Abowd C., Feinstein A., Hmelo C., Kooper R., Long S., Sawhney N. and Tani M.**, Teach and Learning a Multimedia Authoring: The Classroom 2000 project, Proceedings of the ACM Multimedia '96 Conference, 1996
- [Bates94] **Bates J.**, The Role of Emotion in Believable Agents, Communications of the ACM, 37: (7) 122-125 Jul., 1994
- [Bates92] **Bates J.**, The Nature of Characters in Interactive Worlds and the Oz Project, Technical Report CMU-CS-92-200, School of Computer Science, CMU, Pittsburgh, Oct. 1992
- [Bernsen98] **Bernsen N.O.**, Designing Interactive Speech Systems: From First Ideas to User Testing, London, New York: Springer, 1998
- [Bobick98] **Bobick A., Intille S., David J., Baird F., Pinhanez C., Campbell L., Ivanov Y., Schuette A. and Wilson A.**, Design Decisions for Interactive Environments: Evaluating the KidsRoom, Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments, AAAI TR SS-98-02, 1998

- [Brooks85] **Brooks R.**, A Robust Layered Control System for a Mobile Robot, AI Lab Memo 863, Massachusetts Institute of Technology, Cambridge, MA
- [Coen99] **Coen M., Weisman L., Thomas K., and Groh M.**, Context Enhanced Speech Recognition, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1999
- [Coen98] **Coen M., Groh M., Peters S., Phillips B., Weisman L., Warshawsky N., Gajos K., Wilson K., Brooks R. and Lozano-Perez T.**, The Intelligent Room, AI Lab Abstract, 1998
- [Coen97] **Coen M.**, Building Brains for Rooms: Designing Distributed Software Agents, American Association for Artificial Intelligence, 1997
- [Cooperstock97] **Cooperstock J., Fels S., Buxton W. and Smith K.**, Environments: Throwing Away Your Keyboard and Mouse, Communications of the ACM, 1997
- [Dell99] **Dell**, <http://support.dell.com/askdudley>, 1999
- [Don92] **Don A.**, 1992. "Anthropomorphism: From Eliza to Terminator 2", panel description in the Proceedings of the CHI '92 Conference, ACM Press

- [Eliza68] Eliza; A Skimmable Report On the Eliza Conversational Tutoring System, Cambridge, MA, Education Research Center, Massachusetts Institute of Technology, 1968
- [Foner97] **Foner L.**, What's an Agent, Anyway? A Sociological Case Study, The Proceedings of the First International Conference on Autonomous Agents, 1997
- [Gronforst92] **Gronforst S. T.** and **Juhola M.**, Experiments and Comparisons of Inference Methods of Regular Grammars, IEEE Transactions on Systems Man And Cybernetics, 22: 94) 821-826 Jul-Aug, 1992
- [Hajicova92] **Hajicova E.**, Linguistic Aspects of Natural Language Processing, Lecture Notes on Artificial Intelligence, 617: 477-484, 1992
- [Hayden98] **Hayden C.**, <http://chayden.net/chayden/eliza/script> , 1998
- [Hayward68] **Hayward P. R.**, Eliza Scriptwriter's Manual, A Manual for the Use of the Eliza Conversational Computer System, Cambridge, MA, Education Research Center, M.I.T., 1968

- [Horacek97] **Horacek H.**, A Model For Adapting Explanations To The User's Likely Inferences, *User Modeling and User-Adapted Interaction*, 7: (1) 1-55, 1997
- [Ibister98] **Ibister K.** and **Hayes-Roth B.**, Social Interaction with Characters. Submitted to *Journal on Animated Interface Agents*, 1998
- [IE98] Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments, AAAI TR SS-98-02, 1998
- [Katz88] **Katz B.**, Exploiting Lexical Regularities in Designing Natural Language Systems, Cambridge, MA: Lexicon Project, Center for Cognitive Science, MIT, 1988
- [Laurel90] **Laurel B.**, Interface Agents: Metaphors with Character, in the Art of Human-Computer Interaction Design, Ed. Addison-Wesley, Reading, MA 1990
- [Lucente98] **Lucente M.**, **Zwart G.** and **George A.**, Visualization Space: A Testbed for Deviceless Multimodal User Interface, Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments, AAAI TR SS-98-02, 1998

- [Minsky88] **Minsky M.**, The Society of Mind, 1st Touchstone Ed., New York: Simon and Schuster, 1988
- [Mozer98] **Mozer M.**, The Neural Network House: An Environment that Adapts to its Inhabitants, Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments, AAAI TR SS-98-02, 1998
- [Murray99] **Murray J. M.**, <http://web.mit.edu/jhmurray/www/>, 1999
- [Murray98] **Murray J.M.**, Conversational Structure, Improvisation and Lifelike Character, Laboratory for Advanced Technology in the Humanities
- [Murray97] **Murray J.H.**, Hamlet on the Holodeck: The Future of Narrative in Cyberspace, New York: Free Press, 1997
- [Nass94] **Nass C., Steuer J. and Tauber E.**, Computers are Social Actors, in Proceedings of the CHI '94 Conference, Boston, MA, 1994
- [Nuance99] Nuance Communications, <http://www.nuance.com>, 1999
- [Patch98] **Patch K. and Smalley E.**, Globe Correspondents, The Walls Have Eyes - And Ears And ..., The Boston Globe, 07/20/98

- [Phillips98] **Phillips B. and Coen M.**, The Metaglu Agent System, AI Lab Abstract
09/98
- [Picard97] **Picard R.W.**, Affective Computing, Cambridge, MA: MIT Press, 1997
- [Reeves96] **Reeves B. and Nass C.**, The Media Equation: How People Treat
Computers, Televisions and New Media Like Real People and Places,
Cambridge University Press, NY, 1996
- [Reilly96] **Reilly W. S. N.**, Believable Social and Emotional Agents, Ph.D. Thesis,
School of Computer Science, CMU, 1996
- [Rousseau97] **Rousseau D. and Hayes-Roth B.**, Personality in Synthetic Agents,
Technical Report KSL-98-21, Knowledge Systems Lab, Stanford
University, Stanford, CA, July 1997
- [Schank73] **Schank R.C. and Colby K.M.**, Computer Models of Thought and
Language, San Francisco, W.H. Freeman, 1973
- Artificial Intelligence and the Concept of Mind, Allen Newell
 - Semantic Networks: Their Computation and Uses for Understanding English
Sentence, R.F. Simmons
 - An Artificial Intelligence Approach to Machine Translation, Y. Wilks

- A Procedural Model of Language Understanding, T. Winograd
- Identification of Conceptualization Underlying Natural Language, R.C. Schank
- Simulation of Belief Systems, K.M. Colby
- The Structure of Belief Systems, R.P. Abelson
- The Memory We Must Have, E. Hunt
- In Defense of Ad-Hoc Systems, R.K. Lindsay
- A Model for the Encoding of Exp. Information , J.D. Becker

[Shieber93] **Shieber S. M.**, Lessons from a Restricted Turing Test, Communications of the Association for Computing Machinery, volume 37, number 6, pages 70-78, 1994

[Tannen97] **Tannen D.**, Talking From 9 to 5: Men and Women in the Workplace, Avon Books, NY, 1997

[Walker97] **Walker M.A., Cahn J.E. and Whittaker S.J.**, Linguistic Style: Social and Affective Bags of Agent Personality, Proceedings of the First International Conference on Autonomous Agents, pp.96-105, Marina del Rey, CA, Feb. 1997

[Weisman98] **Weisman L. and Muzundar M.D.**, Intelligent Multimodal Drawing Environments, AI Lab Abstract 98

- [Weizenbaum66] **Weizenbaum J.**, ELIZA - A Computer Program For the Study of Natural Language Communication between Man and Machine, Communications of the ACM, vol.9 (1), pp.36-45, 1966
- [Whalen96] **Whalen T.**, Computational Behaviorism Applied to Natural Language, Communications Research Centre, April 30, 1996
- [Winston78] **Winston P.H.**, Learning by .Creating Transfer Frames, Artificial Intelligence, vol.10, no.2, pp.147-72, Netherlands. April 1978
- [Zue92] **Zue V.W.**, Automatic Speech Recognition and Understanding, Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1992ference on Autonomous Agents, pp. 96-105, Marina del Rey, CA, Feb. 1997