BUILDING A USER INTERFACE FOR A LEXICON BROWSER SYSTEM.

JENNIFER SCOTT

A MAJOR REPORT

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 1996

© JENNIFER SCOTT , 1996

National Library of Canada

Bibliothèque nationale du Canada

Acquisitions and Bibliographic Services Branch

Direction des acquisitions et des services bibliographiques

395 Wellington Street
Ottawa, Ontario
K1A 0N4

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-18438-2

Canadä

Abstract

Building a User Interface for a Lexicon Browser System.

Jennifer Scott

The AETNA (Analysis of English Texts from Newspaper Articles) Group at Concordia University is attempting to build a Lexicon with detailed semantic and syntactic information. This report describes the construction of a graphical user interface - The Lexicon Browser System - to be used to help build the AETNA Lexicon. Some of the problems associated with Lexicon building in general and of interface design in particular are described.

Acknowledgments

I would like to thank my supervisor, Sabine Bergler, for all her help and advice during the project, Professor R. Shinghal for his extensive comments, and my husband, Andrew Scott, for his continuing support and patience.

Contents

List of Figures viii

1 Introduction 1

1.1 What is a Lexicon Entry? 1

1.2 The Rationale for Building a Lexicon Browser System 2

1.3 Why Build an Interface for Lexicon Construction? 11

1.4 The Rationale for Interface Design 12

2 Developmental History 14

2.1 The First Prototype 14

2.2 The Second Prototype 15

2.3 The Third Prototype 16

2.4 Combining Two Systems to Make the Lexicon Browser System 18

2.5 The Fourth Prototype 21

3 Using the Lexicon Browser System **27**

3.1 [illegible] of the UIM/X Tool 27

3.1.1 UIM/X Tool 27

3.1.2 Rationale for using UIM/X 28

3.2 Technical Details of the Lexicon Browser System, Buttons, Menus, Windows, etc. 28

3.2.1 Getting Started 28

3.2.2 The Menu Structure 29

3.2.3 FILE 29

3.2.4 DATA MENU 30

3 2.5 STRUCTURE MENU 34

3.2.6 EDIT 35

3.2.7 RESET 35

3.2.8 TEXT BOX 35

3.2.9 SCROLLED WINDOW 35

3.2.10 HELP 36

4 System Evaluation and Suggestions for Future Developments **37**

4.1 System Evaluation 37

4.2 Future Developments 38

5 Conclusion 39

Bibliography 40

List of Figures

1 An example of an on line dictionary. It shows the verb deny. 4

2 An example of a WordNet entry, using the verb deny. 5

3 An example of an AETNA Lexical entry. It shows the verb deny. 7

4 An example of a Noun Hierarchy: taken from WordNet 9

5 The Class Hierarchy for the AETNA Lexicon 10

6 Diagram of the Lexicon Browser Interface. The Third Prototype. 17

7 An Overview of the Lexicon Browser System: showing how the Browser System interacts with Galina's System 19

8 Galina's Lexicon Editor 20

9 An example of a Scheme entry. It shows the verb deny. 21

10 A Schematic View of the Functionality of the Fourth Prototype 22

11 The Fourth Prototype 23

12 The Lexicon Browser System with the File Choice Box open 25

13 The Menu Structure 29

14 The Lexicon Browser System showing a the word [illegible] . . . . . [illegible]

Chapter 1

Introduction

1.1 What is a Lexicon Entry?

A Lexicon entry consists of a headword (the word we are interested in) plus all its associated syntactical and semantical features. These features are arranged in various fields. Each field is built up from a slot name (giving the name of the feature) and a slot value (giving the value ascribed to the feature). The slot names are static and are the same for each word in a particular class of words, but the slot values vary. Some slot values will be inherited from classes higher in the class hierarchy and others will be specific for each word. Entering these specific values makes up the bulk of the work of building a Lexicon.

Syntactic fields, including those that give the part of speech, the forms, the number, and gender of the word, are used to parse sentences. It is not possible to understand the meaning of a sentence without understanding the structure that lies behind it. Other fields are concerned with semantics or word meanings.

In English, one word can have several different word senses, which denote different meanings when it is used in different contexts. Printed dictionaries give lists of these word senses. It is easy for humans to know which word sense is being used in a certain context but very difficult for a machine without our background of world knowledge. In order to lessen this ambiguity, there is a need to minimize the number of word senses in each lexical entry. The entries need to be interrelated according to semantic field and to have a meta-structure that serves different purposes.

The Lexicon should have a dynamic structure and not just the static structure of printed dictionaries. It should be a dynamic structure that derives its power from linking information between lexical entries, from a hierarchical structure, and from a rich meta-lexical structure [Ber95]. Each lexical entry should try to capture the essence, the concept underlying the different word senses. There should also be a way to derive non literal or new word senses from the conventional meaning of a word.

1.2 The Rationale for Building a Lexicon Browser System

In natural language processing, it is important to have large lists of words and their associated syntactic and semantic features (on-line as a Lexicon) for other systems to access. Computational lexicography is concerned with developing computer programs that use machine readable dictionaries in language analysis [BB89]. A machine readable dictionary is an on-line version of a printed dictionary. Syntactic parsing, speech recognition, speech generation, and semantic analysis are natural language processing tasks which could make use of these machine readable dictionaries.

Many groups have built Lexicons for special purposes. Most of these are small (often only a few hundred words) and built with a specific application in mind. Dorr's system called UNITRAN, has about 150 entries and is used for machine translation between English, Spanish, and German [Dor93] and Nirenburg has developed KBMT-89, a system with about 900 entries, to translate between English and Japanese [GN91]. Even though both these Lexicons serve the same purpose, i.e. machine translation, they have very little in common. To enter a new entry, in these two systems, one has to understand the theoretical basis of the user systems. Our Lexicon is designed to have a more general structure and so will be easier to reuse.

There are also several machine readable dictionaries designed for human use that give syntactic information for each word entered e.g. Longman's Dictionary of Current English [Pro78]. These dictionaries have the advantage that they are large databases a lot of the work has already been done. The disadvantage is that they are not structured in a form that can be easily adapted for natural language processing by machine. They were designed to make the printing of dictionaries easier and so contain extra information for this purpose. An example of a typographer's tape from the Collin's English Dictionary [Col79] is shown in Figure 1.

While the Longman's Dictionary of Current English is the most studied machine readable dictionary, the Collin's English Dictionary on CD-ROM was used as an example as this was a machine readable dictionary that our group is able to access [CD-91]. Dictionaries give some of the semantics attached to the word, but not enough to make a reliable Lexicon. They were designed for human users and so can assume the reader has a large amount of background knowledge. They do not contain all the information that is needed if a machine, lacking the human world knowledge, is to use the system.

```
#Hde@.ny #5(d$I#!na$I) #6vb. #1#+nies, #+ny#+ing,
#+nied @n#5(#6tr.#5)@n#1$D. #5to declare (an assertion, statement,
etc.) to be untrue: #6he denied that he had killed her.@n#1$D. #5to
reject as false; refuse to accept or
believe.@n#1$D. #5to withhold; refuse to
give.@n#1$D. #5to refuse to fulfil the requests or
expectations of: #6it is hard to deny a child.@n#1$D. #5to refuse to
acknowledge or recognize; disown; disavow: #6the baron denied his
wicked son.@n#1$D. #5to refuse (oneself) things
desired.@m[C13: from Old French #6denier, #5from Latin
#6d*_eneg*_are, #5from #6neg*_are#5]
```

Figure 1: An example of an on line dictionary. It shows the verb deny.

Printed dictionaries are all organized as lists of lexical entries sorted alphabetically. This makes for limited access to a specific word and makes comparison or generalization across lexical entries a difficult process. WordNet [MBF+90] is an attempt to organize the words differently.

WordNet is an on-line lexical reference system whose design was inspired by current psycholinguistic theories of human memory. It is a network of sense relations. WordNet stores English nouns, verbs, and adjectives organized into synonym sets instead of grouping the words alphabetically as in traditional dictionaries. WordNet contains approximately 54,000 different words organized into about 49,000 word meanings or sets of synonyms [MBF+90]. WordNet does not give enough syntactic or semantic information for it to be used directly as a basis for our Lexicon. It gives information about synonyms and antonyms but is still too limited and does not go far enough for our analysis purposes. Figure 2 shows a typical WordNet entry. WordNet was constructed more for humans to use than for machine reading. The Lexicon being built by the AETNA (Analysis of English Texts from Newspaper Articles) group is to be more detailed in semantic and syntactic information.

```
Antonyms of verb deny

Sense 1
<verb.communication>deny, declare untrue -- ("He denied the allegations")
       <verb.communication>=> admit, acknowledge -- (declare or
                        acknowledge to be true; "He admitted his errors";
                        "She acknowledged that she might have forgotten")


Synonyms (Ordered by Frequency) of verb deny

Sense 1
<verb.communication>deny, declare untrue -- ("He denied the allegations")
                              => contradict, negate, contravene


Hyponyms of verb deny

Sense 1
<verb.communication>deny, declare untrue -- ("He denied the allegations")
       <verb.communication>=> disclaim, make a disclaimer about --
                                     ("He disclaimed any responsibility")
            <verb.communication>=> disavow, refuse to acknowledgel


Hypernyms (Ordered by Frequency) of verb deny

Sense 1
<verb.communication>deny, declare untrue -- ("He denied the allegations")
        <verb.communication>=> contradict, negate, contravene
                   <verb.communication>=> disagree, differ, take issue --
                                 (be at loggerheads; "I beg to differ!")


Sample Sentences of verb deny

Sense 1
<verb.communication>deny, declare untrue -- ("He denied the allegations")
                              *> Somebody ----s something
                              *> Somebody ----s that CLAUSE
```

Figure 2: An example of a WordNet entry, using the verb deny.

The AETNA group has an overall goal of improved information retrieval. The aim is to automatically find information from documents or newspaper articles for a specific purpose. At present many information retrieval systems still rely on matching keywords to find a document or part of a document. The AETNA group is trying to use computational linguistics methods to analyze a text fully and thus to provide the basis to retrieve texts based on more fine-tuned information leading to more accurate retrieval.

The AETNA group has focused on the Newspaper Article domain for two reasons: first, newspapers provide a vast and important fact base with commercial interest and second, there is a lot of cheaply available data that is already on line. Most major newspapers now publish daily on the internet. In this context the proper analysis of reported speech is very important. In order to make a proper analysis of reported speech an improved Lexicon is required.

A useful Lexicon would contain thousands, maybe millions of entries. One method of Lexicon development is to use a machine readable dictionary as a starting point. ASCOT (Automatic Scanning system for Corpus-Oriented Tasks [AMM85]) is a project to develop a computerized Lexicon using Longman's Dictionary of Current English as its base. This is a difficult and complex task and the results seem to be more useful in the syntactic field than in the semantic field that the AETNA group is more concerned with. Also the developers of ASCOT found there was a conflict between the ways information can be made clear to human beings and to computers.

The AETNA project is trying to enrich existing lexical semantics. Therefore the entries are crafted by hand and are structured in a way to be maximally consistent, given the

```
    AETNA Lexical Entry

CLASS <rv>
ROOT:                        deny
FORMS:                       denies denied denying
PART OF SPEECH:              verb
SENSE: 1
 ARGUMENTS: 2
  SUBJECT:                   AGENT
   REQUIRED:                 yes
   SUBCATEGORIZATION:        human
   METONYMIC-EXTENSIONS:     document, company, institution
  OBJECT1:                   theme
   PREPOSITION:
   REQUIRED:                 yes
   SUBCATEGORIZATION:        information
   METONYMIC-EXTENSIONS:
 LEXICAL CONCEPTUAL PARADIGMS
 EVENT-TYPE:                 transition
 TEMPORAL:
 DEFINITION:                 to declare to be untrue
 EXAMPLES:
 HYPERNYMS:
 SYNONYMS:
 ANTONYMS:
 SEMANTIC-CONCEPT:           RV
 LEXICAL CLASS
  VOICE:                     unmarked
  EXPLICIT:                  explicit
  FORMAL:                    unmarked
  AUDIENCE:                  unmarked
  POLARITY:                  negative
  PRESUPPOSITIONS:           presupposed
  SPEECH-ACT:                inform
  AFFECTEDNESS:              unmarked
  STRENGTH:                  unmarked
 DISCOURSE-POLARITY:
```

Figure 3: An example of an AETNA Lexical entry. It shows the verb deny.

known limitations of manual lexicography. Figure 3 shows a typical entry in the AETNA Lexicon. The semantically complex lexical entries have been described in more detail by Sabine Bergler [Ber93] and [Ber95]. This type of entry gives the user access to much more semantic and syntactic information about each word than other on-line dictionaries.

The part of the AETNA project I have been involved with is the building of a user interface for the AETNA Lexicon: The Lexicon Browser System. This interface is a tool to aid in the building and maintenance of the AETNA Lexicon. An easy, efficient method of entering and reviewing data about words, abstracting similarities and differences of meanings related to the word would be very helpful. This Lexicon Browser System is an attempt to do the above. It will be most useful when adding data to the Lexicon.

The Lexicon Browser System will give a user the ability to navigate around potentially thousands of entries. It will be able to display a typical entry to give the user an idea of how to enter details for similar words. Word entries are entered in the Lexicon as templates constructed with many slots. The AETNA Lexicon will be a hierarchy of entries, programmed in the object oriented language TINY-CLOS. Tiny-clos is a variant of CLOS, the Common Lisp Object System [Pae93], embedded in Scheme used to define classes, multiple inheritance, generic functions and primary methods.

The hierarchical structure of printed dictionaries has been shown to be a broad but shallow hierarchy. Amsler [Ams80] in his work with the Merriam-Webster Pocket Dictionary has shown, for example, that noun hierarchies usually have only about 8 levels (maximum 18 levels). Figure 4 shows WordNet's noun hierarchy for different types of chair. This has implications for lexicon data entry. After a stage of building up a basic stock of semantic classes for the upper levels, later data entry would consist of adding entries to the system as leaves.

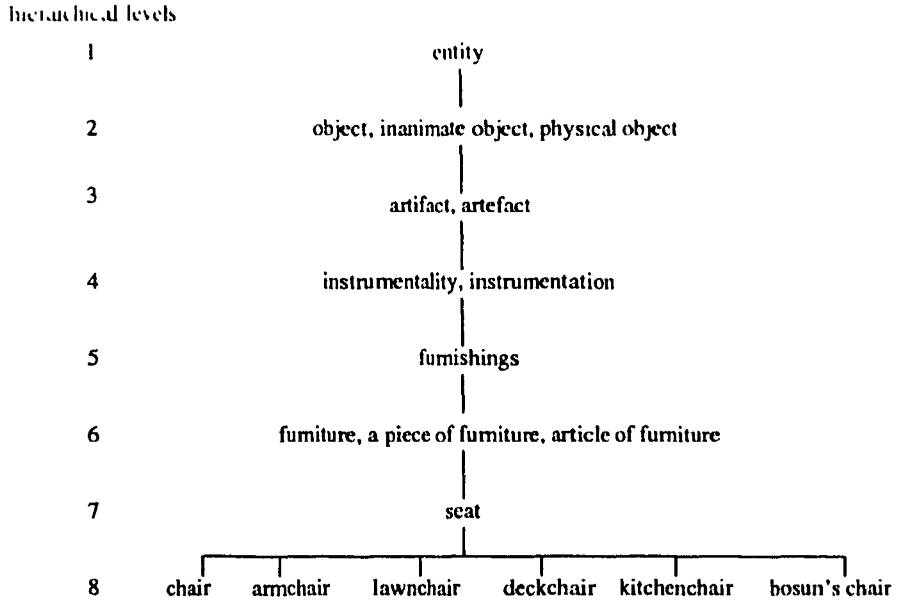
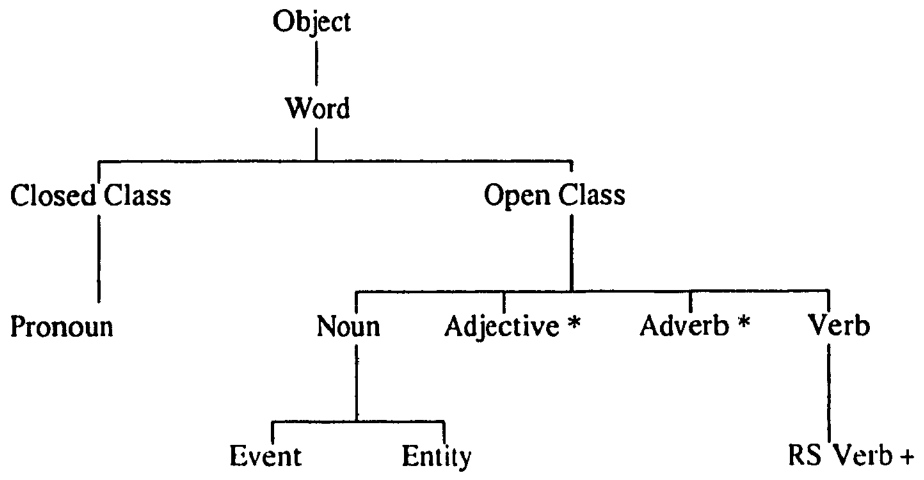

hierarchical levels
1
entity
2
object, inanimate object, physical object
3
artifact, artefact
4
instrumentality, instrumentation
5
furnishings
6
furniture, a piece of furniture, article of furniture
7
seat
8
chair
armchair
lawnchair
deckchair
kitchenchair
bosun's chair

Figure 4: An example of a Noun Hierarchy: taken from WordNet

The dictionaries' hierarchical structure [Ams80] has been made explicit in WordNet. Making use of hyponymy and hypernymy (two semantic relations between word meanings), WordNet creates an inheritance system. Hyponymy / hypernymy is a semantic relation between words and is also called subordination / superordination, subset / superset, or the "isa" relation. For example "maple" is a hyponym of "tree", and "tree" is a hyponym of "plant" - a maple is a kind of tree [MBF+90]. A hyponym inherits all the features of the more generic concept and adds at least one feature that distinguishes it from its superordinate and from any other hyponyms of that superordinate [MBF+90].

It is important to allow for related words to inherit default slot values. (e.g. a "musician" is a kind of "human" and so the word "musician" should inherit human characteristics. A "violinist" should inherit characteristics from both "musician" and "human".) Allowing new words to inherit default slot values would simplify and lessen the task of data entry. It would cut down on repetition and would help in keeping entries accurate. CLOS [Pae93], an object oriented language, is being used to make building the hierarchy and incorporating inheritance easier. Figure 5 shows the proposed class hierarchy to be used in the AETNA Lexicon.



* These Templates have not been developed yet

+ For the definition of RS Verb see Chapter 3, Section 3.2.4

Figure 5: The Class Hierarchy for the AETNA Lexicon

The system should also facilitate checking entries for accuracy and consistency. Using an object oriented hierarchy with inheritance should make this checking easier. The Browser should give the user an alphabetically sorted list of words already entered in the Lexicon and should show the user specific words with all their related semantic and syntactic details. It should show details of the different senses of the same word.

The only users of the system are at present members of the AETNA group, but if the project is successful the Lexicon Browser System could be used by others interested in building a lexicon for computational purposes.

1.3 Why Build an Interface for Lexicon Construction?

There are several problems associated with data entry. Firstly, the AETNA group is still in the process of deriving the format of the entries for our Lexicon. This means the system must be flexible enough to change as the style and contents of individual entries change.

Secondly, the size of the finished Lexicon poses many problems. It is hard to develop a small accurate system, but it is very much harder to scale up a small system to one that can cope with thousands of words. It will take many man hours of work to complete the task. The system must be robust enough to cope with many different people, with varying levels of computer literacy, working on data entry. Data entry is time consuming, the development of a regular dictionary involves tens of lexicographer-years [BB89].

Thirdly, data entry is a task where it is easy for the human to make mistakes. The Lexicon interface should provide for some automatic spell checking, checking for data consistency, and should allow the user to view entries for similar words so that the user can get the format correct.

In this project the language used for the application is SCHEME. The tool for graphical interface development is UIM/X. The object oriented hierarchy is being implemented with TINY CLOS. The reason for these choices is to use Public Domain software. This is to make the Lexicon and Lexicon tools easily portable so that they can eventually be used by as many people as possible.

1.4 The Rationale for Interface Design

Good interface design is very important for the quality of the finished product in software engineering. The most important factor affecting productivity is the complexity of the user interface. A simple interf. ce is very much easier for the user to understand and to learn to use. It is very hard to define what is a good or bad interface as this involves subjective judgment. A very good application can fail if the user is irritated with the user interface, and a mediocre application may succeed with a good user friendly interface. The interface is the only part of an application the user interacts with and so it contributes a great deal to the ultimate success of a product [Som89, pages 489 - 491].

A badly designed interface can cause the user to misunderstand the meaning of a presented item leading to mistakes. An interface that is hard to understand and hard to use will result in a much higher error rate when the user is entering data. The interface must be designed with the needs of the user in mind. The components of the interface should be those the user is already familiar with and actions associated with the components should be consistent.

Interface design is often an afterthought tacked on to the end of product development. This often leads to a bad interface. In this project the interface is being developed in parallel with the rest of the application and has changed as the needs of the project have changed.

Colour can be used for emphasis in interface design and to make an interface more interesting to the user. Too many colours should not be used as this makes an interface confusing and more complicated. The use of colour should be consistent [Som89, pages 283 - 285].

The interface has been designed in colour but works as well in monochrome. Colour has been used to highlight parts of the display and to distinguish between different areas of the windows. Colour has not been used to convey meaning as the user may only have a monochrome monitor or the user may be colour blind.

Chapter 2

Developmental History

2.1 The First Prototype

The Lexicon Browser System was started as a simple command line application written in scheme on an IBM compatible PC. The first prototype was developed using Texas Instruments PC Scheme, briefly referred to as TI Scheme.

This was a small version of the Lexicon system with routines to add words, delete words, show word details, show a list of all words in a list of mock entries called the Lexicon, save words and to do the same with reported speech verb entries. The LISP and Scheme data type "structure" was used to implement reported speech verb templates and slot values.

Choices were made from various menus using the keyboard. The letter entered from the keyboard called up the next menu level or showed the required entry.

This first prototype could handle only a very small, simulated Lexicon. The screen was small and parts of the entries for the reported speech verbs would scroll off the screen. For a lexicon of thousands of words, as is planned this system would take too long to load to sort entries, and to search for words.

The templates, slots and slot fillers that go to make up a reported speech verb entry were bound together using structures. There was a problem with this because TI Scheme is limited to 32 of these slots. This was a drawback as each reported speech verb requires more slots than this to fully characterize each verb.

The first prototype helped to show what was possible and what would be needed later. A more flexible user interface was required. An interface with windows and mouse capabilities would allow the user to scroll up and down the long entries and lists of words in the lexicon. The availability of more than one window would let the user view a completed entry while adding to or developing similar entries. For the above reasons the system was ported to the SUN UNIX System.

2.2 The Second Prototype

This was the Lexicon Browser System running on UNIX. It was necessary to change from TI Scheme to Gambit to get the system to work on the UNIX system. Gambit was chosen over MIT Scheme, both are Public Domain software. Gambit is an IEEE and R4RS-conformant small implementation of Scheme [KR92], which is faster, runs on different platforms, and has both an interpreter and a compiler. Scheme is a language that has a formal definition of both syntax and semantics.

Several alterations to the Scheme code were necessary at this stage. The most difficult problem was that Gambit does not provide a procedure to read in structures from a file. The second prototype was still a command line application.

2.3 The Third Prototype

For both the first and second prototypes a simulated Lexicon had been constructed to simplify the task. Now it appeared that the simplification was causing most of the problems and the third prototype was a step to interface ultimately with the tools for real Lexicon storage and maintenance already developed.

It was decided to develop a more user friendly interface with windows and mouse capabilities. To do this it was decided to use the UIM/X Graphical User Interface Tool. UIM/X is a complex tool and to get to know it I followed the series of 9 tutorials provided by Visual Edge Software Ltd. in their "Getting started with UIM/X" package [Vis93a].

The third prototype consisted of a small UIM/X interface that worked with the second prototype. When the application was called, a bulletin board containing a scrolled window popped up on the screen. Push buttons activated pull down menus to allow the user to interact with the application. Messages were sent to the Gambit subprocess and results were shown in the scrolled window. A diagram of the third prototype is shown in Figure 6.

The command line menus were converted to pull down menus that could be used with a mouse. In order to get the third prototype to work, the Gambit code had to be modified again. Less Scheme code was now needed as the menus were implemented by the graphical interface.

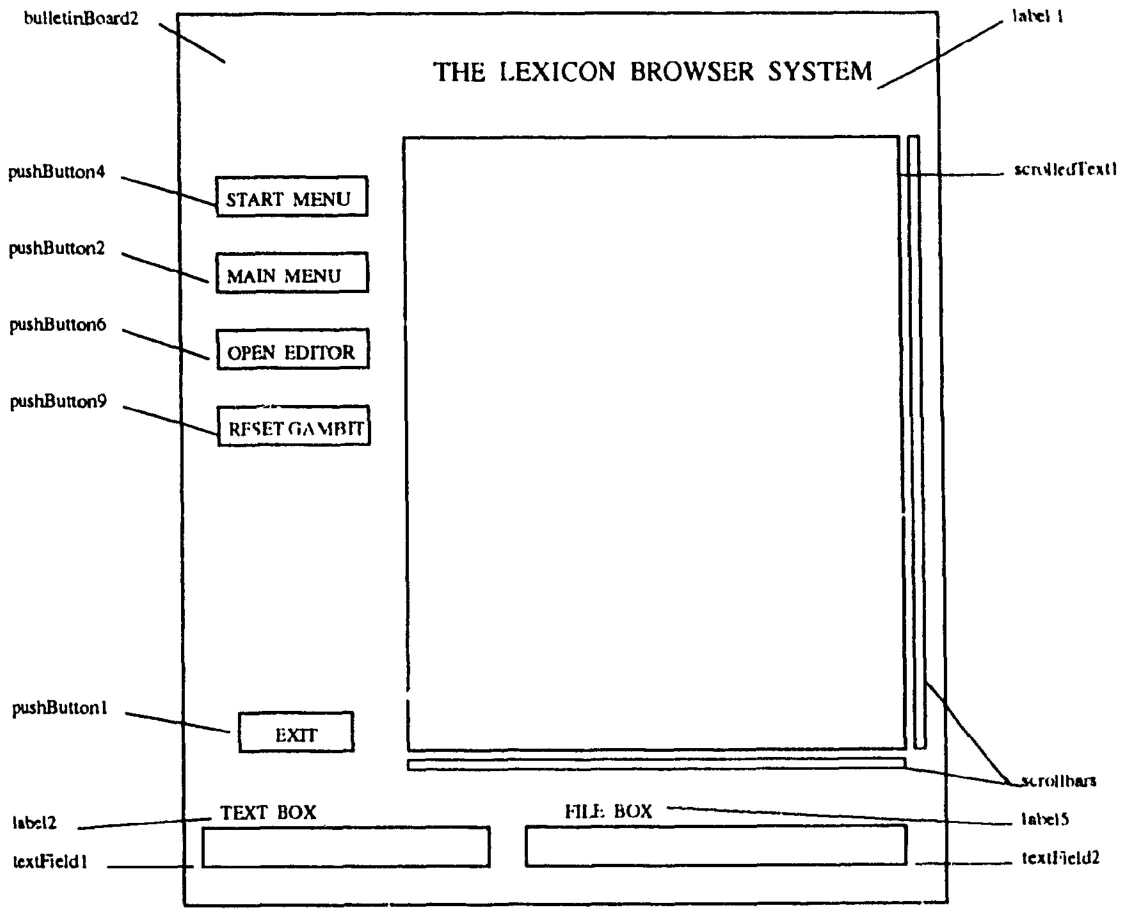


Figure 6: Diagram of the Lexicon Browser interface. The Third Prototype.

2.4 Combining Two Systems to Make the Lexicon Browser System

At this stage it was necessary to use code developed by Galina Kolesova (Systems Analyst in the Department of Computer Science, Concordia University) to manage the full size Lexicon. It would not be a simple matter to scale up my system to at one point manage a Lexicon of several thousands of large entries. Galina developed a system, Galina's Lexicon Editor, to manage editing, saving and sorting functions. This required complex management strategies and the use of hash tables.

As explained previously, words need to be placed in a hierarchy so that related words can inherit slot values. Galina is working in Tiny Clos with its object orientation to implement this

The Lexicon Browser System was developed by combining ideas developed in the third prototype with Galina's system. The third prototype allowed the user to open the Galina's Lexicon Editor window to edit the Lexicon entries. The relationship between the two systems can be seen in Figure 7.

This was a difficult stage as it was hard to work with code written by two people. It later required a major revision into the fourth prototype as there was still some conflict between the two subsystems: they did not communicate as well as they should have.

The third prototype allowed the user to open files necessary to start Gambit and to start the Browser system. The user then used the Main Menu to review previous entries (verbs and reported speech verbs, only as nouns are not finished yet). The user could use

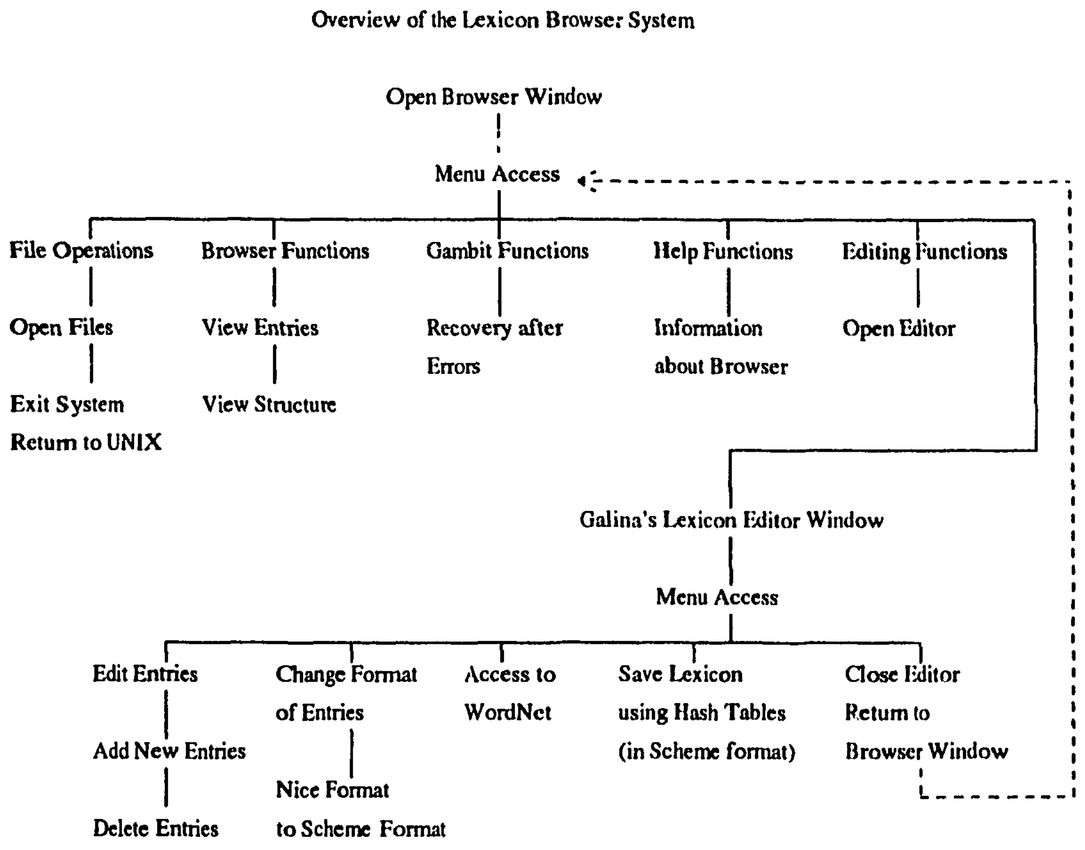


Figure 7: An Overview of the Lexicon Browser System: showing how the Browser System interacts with Galina's System

the Text Box to enter words to check whether they were already in the Lexicon. The File Box allowed the user to enter the full path name of the Lexicon file he was interested in. When the user wished to edit or save work done on an entry, the Open Editor button gave access to Galina's Lexicon Editor. The user ended a session with the Exit button.

The Lexicon Editor developed by Galina is shown in Figure 8.

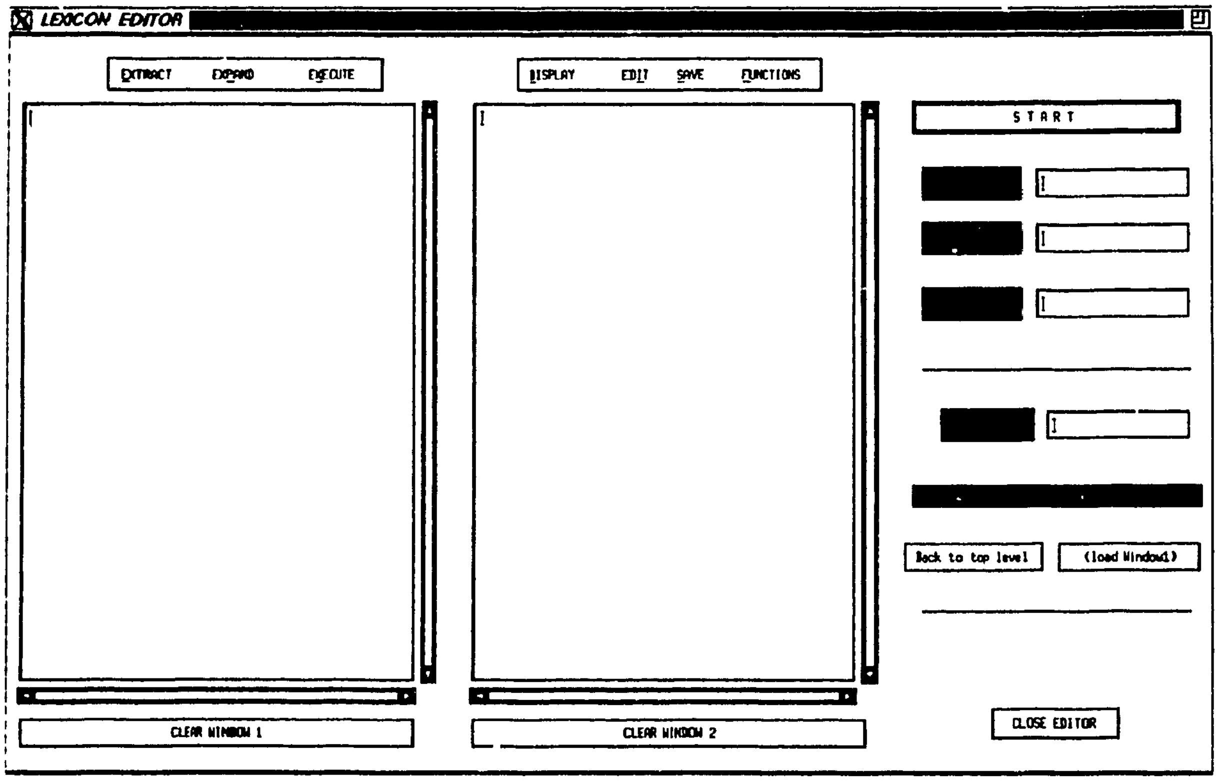


Figure 8: Galina's Lexicon Editor

Galina's Lexicon Editor allows the user to perform the following useful tasks associated with entry management. The user can add, delete, edit, and save entries. Entries can be added to the Lexicon either by typing data directly onto the screen in a window or by reading an entry from a file in the "nice" format. The "nice" format is a form of the Lexicon entry that is easily readable and understandable by the user whereas the "scheme" format is a more machine readable format. The Lexicon Editor can automatically convert from the "nice" format to the "scheme" format. The entries are saved in the "scheme" format. Figures 9 and 3 show examples of entries in the "scheme" and the "nice" formats.

Galina's Lexicon Editor also provides the user with the option of automatically converting WordNet entries into the slot-filler structures used for Lexicon entries which can then be edited and saved in the Lexicon files. The user can access words via the root of the word.

```
(make-obj <rv> "deny"
 'FORMS "denies denied denying"
 'POS "verb"
 'SENSE "1"
 'args (list  '()(list 'THETA-ROLE "theme" 'REQ "yes" 'SUB
CAT "information" ))
 'DEFINITION "to declare to be untrue"
 'SEMANTIC-CONCEPT "RV"
 'POLARITY "negative"
 'PRESUPPOSITIONS "presupposed"
)
(make-obj <rv> "deny"
 'FORMS "denies denied denying"
 'POS "verb"
 'SENSE "2"
 'args (list  '()(list 'THETA-ROLE "SCOMP" 'PREP "optional
 that  scomp ''that''" 'REQ "yes" 'SUBCAT "information" ))
 'DEFINITION "to declare to be untrue"
 'SEMANTIC-CONCEPT "RV"
 'POLARITY "negative"
 'PRESUPPOSITIONS "presupposed"
)
```

Figure 9: An example of a Scheme entry. It shows the verb deny.

2.5 The Fourth Prototype

For the fourth and final prototype, the system has been revised to make the interface more attractive and easier to use. The menus have been totally redesigned to be more logical and user friendly. The conflicts between the two subsystems have been resolved and the system as a whole works better. An overview of functionality of the fourth prototype can be seen in Figure 10.

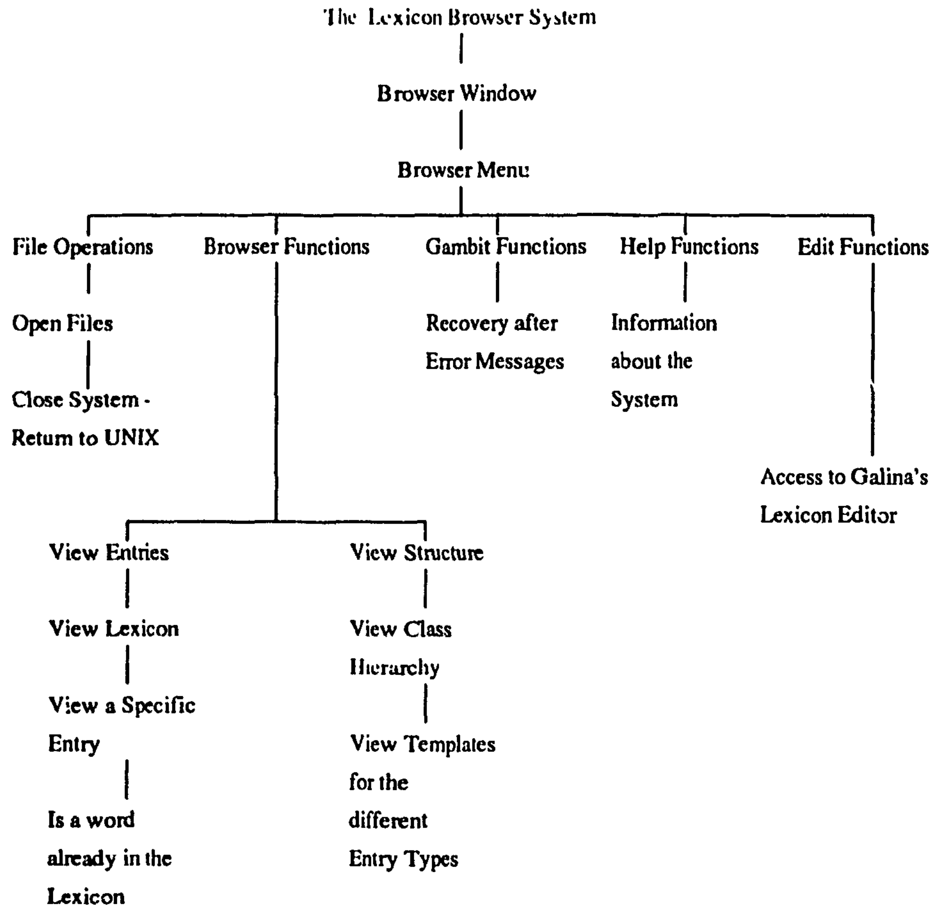
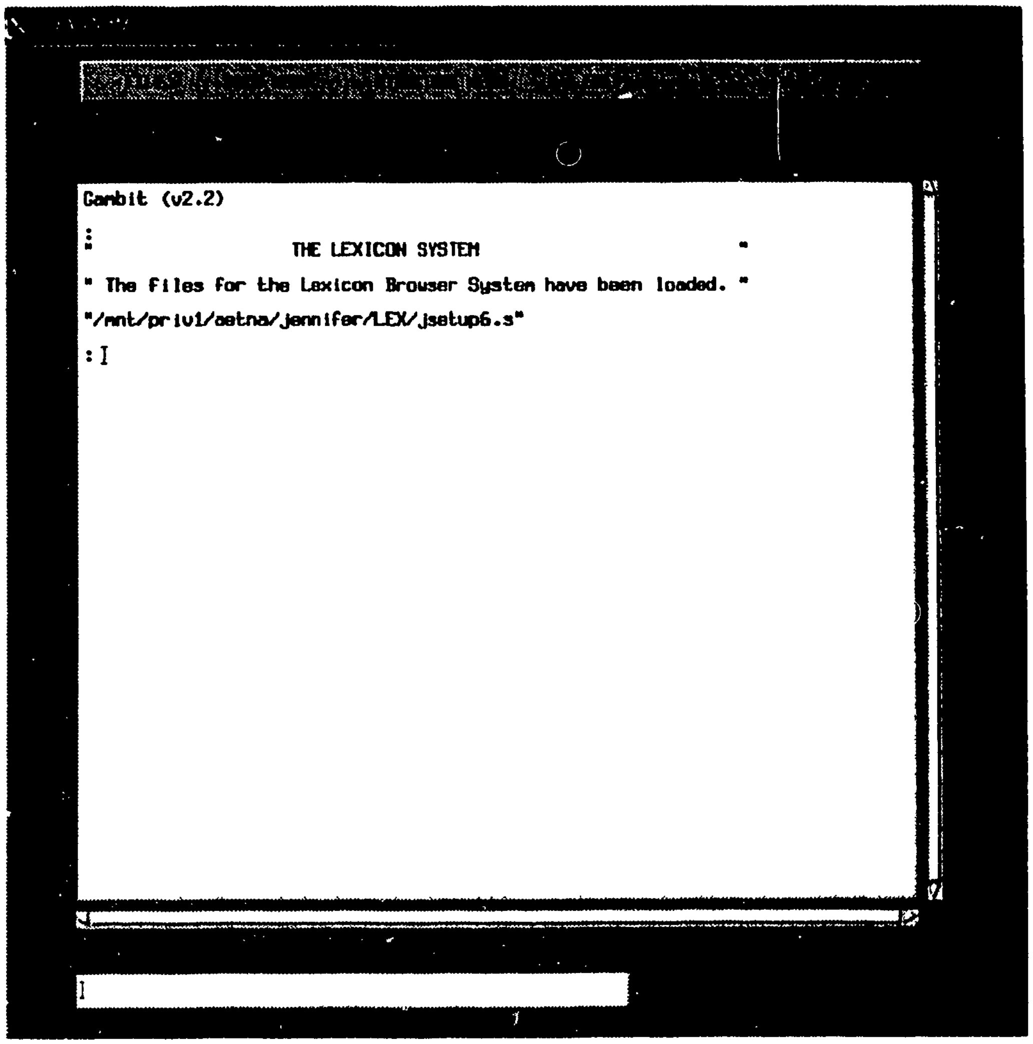

The Lexicon Browser System
Browser Window
Browser Menu
File Operations
Browser Functions
Gambit Functions
Help Functions
Edit Functions
Open Files
Close System - Return to UNIX
Recovery after Error Messages
Information about the System
Access to Galina's Lexicon Editor
View Entries
View Structure
View Lexicon
View a Specific Entry
Is a word already in the Lexicon
View Class Hierarchy
View Templates for the different Entry Types

Figure 10: A Schematic View of the Functionality of the Fourth Prototype

The fourth prototype allows the user to review previous entries, to check whether a word has already been entered in the Lexicon, and to review template structures. It allows the user to make choices between the various Lexicon files to work on. The user can now see which files are available rather than having to remember this. When the user needs to save or edit an entry there is access to Galina's Lexicon Editor by using the Open Editor button.

The system now loads Gambit automatically. A view of the system as it opens can



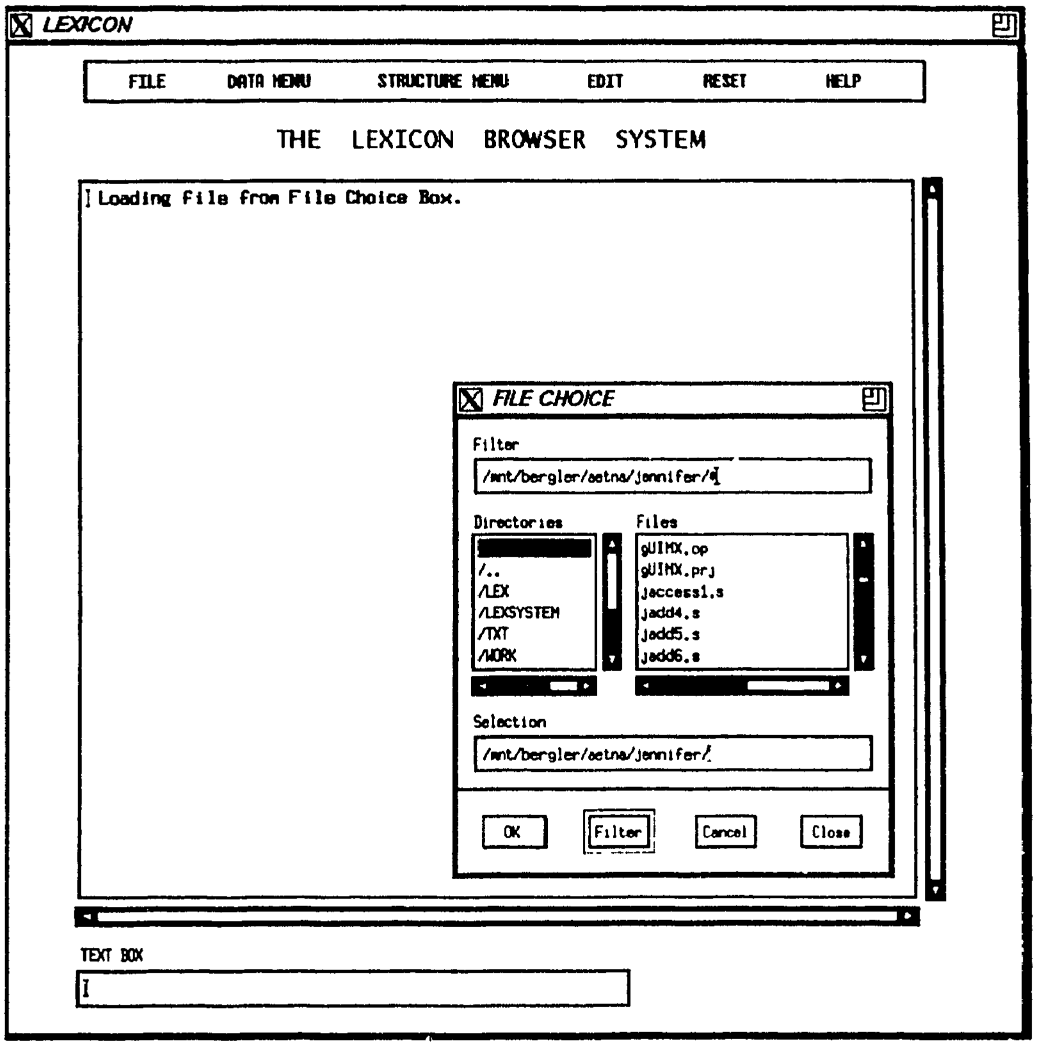
```
Gambit (v2.2)
:
"                    THE LEXICON SYSTEM                     "
" The files for the Lexicon Browser System have been loaded. "
"/mnt/privl/aetna/jennifer/LEX/jsetup6.s"
:
```

Figure 11: The Fourth Prototype

be seen in Figure 11. This was previously done by the user having to click on the START GAMBIT button after selecting the START MENU. As this was something the user needed to do before using the rest of the system, it made more sense to initialize this for the user. The system now also loads the several files needed to start the application. This was changed for the same reasons the method of loading Gambit was changed. The user will see a message in the scrolled window if the system cannot find or load the necessary files.

The FILE BOX where the user had to enter the path names for lexicon files to be loaded (by clicking on a LOAD FILE push-button) has been removed in the fourth prototype. It has been replaced by a FILE CHOICE BOX opened on the screen by clicking on OPEN FILE in the FILE menu. The FILE CHOICE BOX shows the user all the files available. There is a filter mechanism to show the user files in different directories as well. It is easier for the user to make a choice of files from those displayed to remember all the details of a complex path name. Once the relevant file has been selected, a click on the OK button loads the file. An easy method for loading files is essential as the final lexicon will be stored using many files (e.g. at least one file for each letter in the alphabet). Figure 12 shows the Browser System with the File Choice Box open.

The menu system has been pruned and simplified replacing the third prototype version. The number of menu items has been reduced. The menu has been changed from a system of push buttons clicked on with the left mouse button and pull down menus activated with the right mouse button to an easier to use menu bar with pop down menus activated using only the left mouse button. The system is now more consistent with what happens when other commercial windows systems are used. It is now easier to follow and the duplications of the previous version have been removed. Every push button at the end of each menu



```
Loading File from File Choice Box.
```

```
/mnt/bergler/aetna/jennifer/*
```

```
/..
/LEX
/LEXSYSTEM
/TXT
/WORK
```

```
gUIMX.op
gUIMX.prj
jaccess1.s
jadd4.s
jadd5.s
jadd6.s
```

```
/mnt/bergler/aetna/jennifer/
```

Figure 12: The Lexicon Browser System with the File Choice Box open

path now have an effect when clicked, even if it is only to give the user a message - as in the case of nouns - that this part of the system has not been developed yet. (The structure of nouns is still being discussed by the AETNA group).

The EXIT function no longer has a separate menu button and is incorporated into the FILE menu as this is where it is found in most other windows applications. It is better to give the user a familiar pattern so that it is easier to learn how to use the system.

A HELP menu has been added on the far right of the menu bar. Again this is the most common position for help buttons. The help system is at present limited to short descriptions of the functions of the other buttons, but it should not be too difficult to add to this later.

Chapter 3

Using the Lexicon Browser System

3.1 Description of the UIM/X Tool

3.1.1 UIM/X Tool

UIM/X is a Graphic User Interface builder developed by Visual Edge Software Ltd., Quebec. It was produced to allow software developers to interactively create, test, and generate code for user interfaces. Graphical interfaces can also be created for keyboard applications with little modification of the existing applications.

UIM/X is integrated with the OSF/Motif tool kit. Many add-in features called widgets such as push buttons, scroll bars, popup menus are available for designing user interfaces. It provides for the use of multiple windows and interactions with a mouse.

UIM/X contains a C interpreter. There is a large library of working UIM/X functions. UIM/X will automatically generate error free C or C++ code to go with the widgets. It will also generate a customizable main program.

3.1.2 Rationale for using UIM/X

UIM/X was chosen as a development tool because it was available on the UNIX SUN system and to use such a tool would make development of the Lexicon interface quicker and easier. UIM/X provides a comprehensive tutorial "Getting started with UIM/X" [Vis93a] and the "UIM/X Developer's Guide" [Vis93b] to help the novice developer. It is a complex tool and takes quite a long time to understand and make use of its full potential, but it would have taken very much longer and been much more difficult to get to the same stage from scratch.

The menus, push buttons, and scroll bars and other widgets are standard and so familiar to the user. This means the interface is easily understandable and the user should need little learning time. The interface is more user friendly than one produced from non standard parts.

3.2 Technical Details of the Lexicon Browser System, Buttons, Menus, Windows, etc.

3.2.1 Getting Started

To start the Lexicon Browser System the user types LEXBS [enter] at the UNIX prompt.

The main Lexicon Browser System interface then appears on the screen after a short pause as shown in Figure 11. The interface consists of the main Scrolled Window for system/user communication, the Menu Bar at the top of the interface, and the Text Box used to enter data at the bottom of the interface.

The Lexicon Browser System interface has a light blue background. The Menu Bar and pull down menus have black text on a light gray background. The main Scrolled Window and the Text Box show bold black text on a white background. Blue and grey are colours that are restful to the eyes and serve to emphasize and contrast the working areas with the white backgrounds. The system works as well using monochrome monitors where the text is black on a white background.

3.2.2 The Menu Structure

The menu structure is as shown in Figure 13.

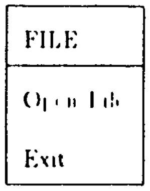
FILE	DATA MENU	STRUCTURE MENU	EDIT	RESET	HELP
OPEN FILE	VIEW ALL WORDS	VIEW HIERARCHY	OPEN EDITOR	RESET GAMBIT	FILE
EXIT	FIND A WORD	VIEW TEMPLATES			DATA MENU
	VIEW AN ENTRY				STRUCTURE MENU
					EDIT
					RESET
					TEXT BOX

Figure 13: The Menu Structure

The items in the menu are set out in the order that the user is most likely to use them.

3.2.3 FILE

Clicking on FILE with the left mouse button displays the following pull down menu:



OPEN FILE

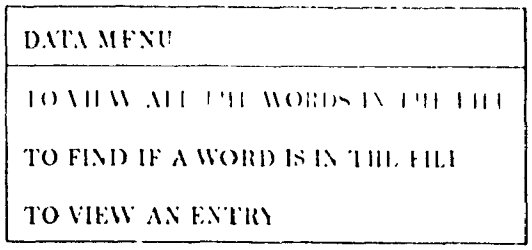
Clicking on OPEN FILE pops up the FILE CHOICE BOX interface on the screen. This allows the user to look through the available files and choose a Lexicon file to load. It shows the user all the files in the directory and the filter option lets the user find files in other directories. A Lexicon file should be chosen and loaded before the user starts browsing through entries.

EXIT

If this button is clicked on with the mouse, the Lexicon Browser System will be closed. The user is returned to the UNIX prompt. The EXIT button does not save any changed or added entries to a file. To do this, the user should use the SAVE function in Galina's Lexicon Editor Window.

3.2.4 DATA MENU

Clicking on DATA MENU displays a series of menus used to browse though the entries in the Lexicon. Clicking on DATA MENU with the left mouse button displays the following pull-down menu:

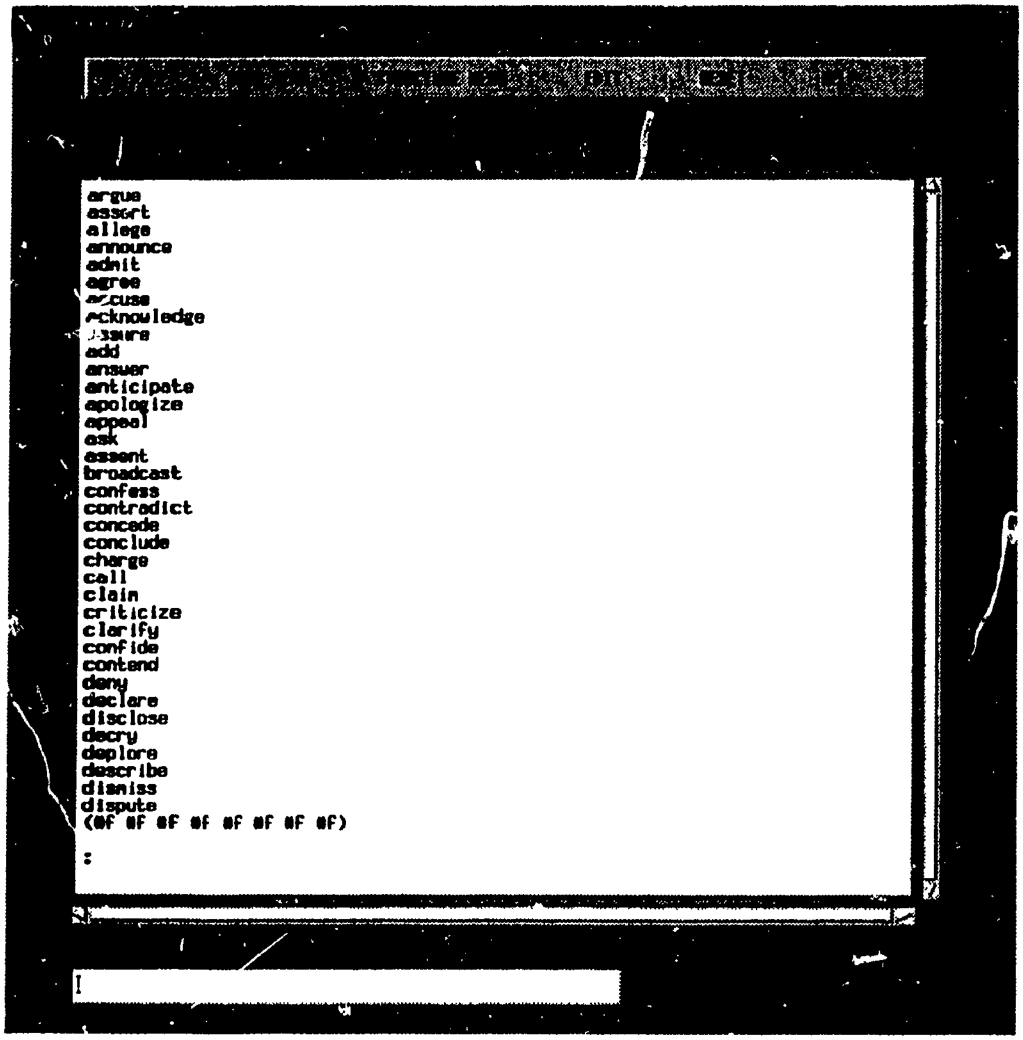

DATA MENU
TO VIEW ALL THE WORDS IN THE FILE
TO FIND IF A WORD IS IN THE FILE
TO VIEW AN ENTRY

TO VIEW ALL WORDS IN THE FILE

This menu button displays a list of all the words in of the Lexicon in that particular Lexicon file. Figure 14 shows the Lexicon Browser System displaying all the words entered in the file HashTable.scm. The words are accessed by the root and the display can be scrolled up and down using the scroll bars attached to the Scrolled Window. This button currently does not interact with the real Lexicon files and uses temporary files instead, since the real Lexicon has not been developed yet. The files used to display entries have to be loaded using the OPEN FILE button in the FILE menu.

TO FIND IF A WORD IS IN THE FILE

If a word is typed into the Text Box, this function will check if the word is present in that particular Lexicon file. If the word is present the Browser will say that it is present; if it is not, a message to say that it is not present appears in the Scrolled Window. If there is no word in the Text Box, a message asking the user to type in a word appears in the Scrolled Window.



```
argue
assert
allege
announce
admit
agree
accuse
acknowledge
assure
add
answer
anticipate
apologize
appeal
ask
assent
broadcast
confess
contradict
concede
conclude
charge
call
claim
criticize
clarify
confide
contend
deny
declare
disclose
decry
deplore
describe
dismiss
dispute
(#F #F #F #F #F #F #F #F)

:
```

Figure 14: The Lexicon Browser System showing all the words in a file.

TO VIEW AN ENTRY

This cascade button shows the following menu items:

TO VIEW AN ENTRY
View a Verb Entry
View a RS Verb Entry
View A Noun Entry

Notice that for the work being done by the AETNA group, it is important to distinguish semantic as well as syntactic categories. The class - RS Verb - denotes the class of reported speech verbs, a subclass of verbs.

TO VIEW A VERB ENTRY

This uses a Verb entered in the Text Box to display the entry in the Scrolled Window. If there is no Verb in the Text Box a message appears in the Scrolled Window asking the user to type one into the box.

TO VIEW A RS VERB ENTRY

This uses a RS Verb entered in the Text Box to display the entry in the Scrolled Window. If there is no RS Verb in the Text Box a message appears in the Scrolled Window asking the user to type one into the box.

TO VIEW A NOUN ENTRY

A message in the Scrolled Window tells the user that nouns are not available yet. Noun entries are still being developed. Other word types will be added later (adjectives etc.).

3.2.5 STRUCTURE MENU

Clicking on STRUCTURE MENU allows the user to browse through the structure of the Lexicon. It lets the user view the word templates and shows the class hierarchy. Clicking on STRUCTURE MENU with the left mouse button displays the following pull-down menu:

STRUCTURE MENU
TO VIEW THE CLASS HIERARCHY
TO VIEW TEMPLATES

TO VIEW THE CLASS HIERARCHY

Clicking on this menu button shows the Class Hierarchy in diagrammatic form. The hierarchy is not complete. More word classes will be added to the diagram as they are developed.

TO VIEW TEMPLATES

Templates show the user which slots are defined for entries of each type. This is to be used when making a new entry. The user can view the template in the Scrolled Window while editing an entry in Galina's Lexicon Editor Window. This cascade button shows the following menu items:

TO VIEW TEMPLATES
View a Verb Template
View a RS Verb Template
View a Noun Template

TO VIEW A VERB TEMPLATE

When this menu button is clicked on the Scrolled Window shows the structure of a verb template.

TO VIEW A RS VERB TEMPLATE

Clicking on this menu button gives the user a RS Verb template in the Scrolled Window.

TO VIEW A NOUN TEMPLATE

A message in the Scrolled Window tells the user that this has not yet been completed.

3.2.6 EDIT

Clicking on the OPEN EDITOR push-button pops up Galina's Lexicon Editor Window. This is the part of the Browser System where editing - including adding, deleting, saving - entries can be performed.

3.2.7 RESET

The RESET GAMBIT push-button is used to return Gambit to its top level after an error message.

3.2.8 TEXT BOX

This is the area to enter words to be accessed in the Lexicon.

3.2.9 SCROLLED WINDOW

This is the area of the interface where messages are shown and the entries of the Lexicon are displayed. Long lists or entries can be scrolled through by using the horizontal and vertical scroll bars attached to the Scrolled Window.

3.2.10 HELP

When the mouse is clicked on the HELP button, the following menu choices appear:

HELP
FILE
DATA MENU
STRUCTURE MENU
EDIT
RESET
TEXT BOX

When an item in the above list is highlighted and clicked on, a short explanation of the function of each widget pops up in the Scrolled Window.

Chapter 4

System Evaluation and Suggestions for Future Developments

4.1 System Evaluation

The Lexicon Browser System has not been formally tested, but at each stage in the design I have been in consultation with the users. Members of the AETNA group have tried the interface and made suggestions for improvements which were incorporated into the later prototypes.

The interface has been checked to make sure what is expected to occur does occur when each item or push-button is clicked on and that the actions produced by each button are consistent.

When the whole system is ready for the start of serious data entry, the actual people who are to enter the data should be observed entering sample words. Their opinions of the system and complaints about the system should be noted and improvements should be added then. In a large task such as Lexicon building, ease of data entry and consistency of entry are very important.

4.2 Future Developments

At present a few verbs and reported speech verbs are the only classes of words with their lexical structures developed. When the noun templates have been developed, actions and behavior will need to be added to the relevant buttons and menu items to enable the user to look up noun details in a similar fashion to reported speech verbs. Following nouns, the development of templates for pronouns, adverbs, and adjectives are envisioned. When the final structure and inheritance patterns of the different word classes has been worked out, this can be added to the interface.

When we have a real Lexicon with hash tables for looking up words, menu items such as VIEW ALL WORDS IN THE FILE will be changed to VIEW ALL WORDS IN THE LEXICON and the behaviour added here will look the words up in the hash tables rather than just looking for them in the chosen Lexicon file.

Other windows or interfaces can be added to this application. Later in the development of this tool, Galina's Lexicon Editor Window could be removed and exchanged for a new and improved version without too much difficulty. This is at present under development by Galina.

Chapter 5

Conclusion

The Lexicon Browser System interface is now working well and should make it easier for AETNA group members to enter the large amounts of data necessary to complete the project of building a meaningful Lexicon.

I have gained insight into the use of a GUI building tool, LATEX, and XFIG while working on this project.

The endeavor of designing and developing a Lexicon for general use is an ongoing project; the tool is therefore not complete. Other members of the AETNA group are still researching and discussing the best possible lexical entry structures. At present the reporting speech verbs are the class of words that have been studied the most. The interface I have designed is flexible enough to change and improve as the needs of the group change. Building a usable Lexicon is a very large and complex task and having a graphical user interface will make this task a little easier.

I now understand more about how complex and difficult natural language processing really is and how large and complicated a problem building up a meaningful Lexicon will be.

Bibliography

[AMM85] Erik Akkerman, Pieter C. Masereeuw, and Willem J. Meijs. *Designing a Computerized Lexicon for Linguistic Purposes.* Rodopi, Amsterdam, 1985.

[Ams80] R. A. Amsler. *The Structure of the Merriam-Webster Pocket Dictionary.* PhD thesis, University of Texas, 1980.

[BB89] Bran Boguraev and Ted Briscoe, editors. *Computational Lexicography for Natural Language Processing.* Longman, London and New York, 1989.

[Ber93] Sabine Bergler. Semantic dimensions in the field of reporting verbs. In *Making Sense of Words, Proceedings of the Ninth Annual Conference of the UW Centre for the New OED and Text Research*, Oxford, 1993.

[Ber95] Sabine Bergler. Generative lexicon principles for machine translation: A case for meta-lexical structure. *Machine Translation*, 9:155–182, 1995.

[CD-91] Association for computational linguistics: Data collection initiative CD-ROM 1, 1991.

[Col79] William Collins, editor. *Collins English Dictionary.* William Collins Sons and Co. Ltd., 1979.

[Dor93] Bonnie J. Dorr. Interlingual machine translation: a parameterized approach. *Artificial Intelligence*, 63, 1993. Special Volume on Natural Language Processing.

[GN91] K. Goodman and S. Nirenburg. *The KMBT Project: A Case Study in Knowledge-Based Machine Translation.* Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[KR92] William Klinger and Jonathan Rees. The Rivised^4 Report on the Algorithmic Language Scheme. *LISP Pointers IV*, 3:1–55, 1992.

[MBF+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Intoduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[Pae93] Andreas Paepke, editor. *Object-oriented programming: the CLOS perspective.* MIT Press, Cambridge, Mass., 1993.

[Pro78] P. Procter, editor. *Longman Dictionary of Contempory English.* Longman, London and New York, 1978.

[Som89] Ian Sommerville. *Software Engineering.* Addison-Wesley Publishing Company, 1989.

[Vis93a] Visual Edge Software Ltd., Quebec, Canada. *Getting Started with UIM/X.*, 1993.

[Vis93b] Visual Edge Software Ltd., Quebec, Canada. *The UIM/X Developer's Guide.*, 1993.