Glosser-R$\underline{u}$G; Prototype December 1996

D.A.Dokter
Alfa-Informatica, University of Groningen
Postbus 716, 9700 AS, Groningen
dokter@let.rug.nl

July 10, 1997

Abstract

This report describes a prototype of Glosser-R$\underline{u}$G that was released in December 1996. Glosser-R$\underline{u}$G is a *Computer Assisted Language Learning* (CALL) application that facilitates the reading of French texts for native Dutch speakers, containing new or unfamiliar words. The application supports this task by providing an on-line dictionary, morphological analysis, part-of-speech disambiguation and examples of word use in a corpus of texts. Hence, the application is both useful in language learning and as a practical tool for reading texts in a foreign language. This report consists of a short overview of the theoretical background of the project, a complete description of the program's front-end, underlying mechanisms and implementational issues, an example session, and a comparison between this prototype and previous versions of Glosser-R$\underline{u}$G.

1 The Position of Glosser-R$\underline{u}$G in Computer Assisted Language Learning

The development and research of Glosser-R$\underline{u}$G is incorporated within the larger framework of the GLOSSER project, which aims to apply state-of-the-art natural language processing techniques to technology for *Computer Assisted Language Learning* (CALL) [9]. The specific goal of Glosser-R$\underline{u}$G within this project is facilitating the reading of French texts for native Dutch speakers. In this respect the focus of the application in the CALL paradigm is clearly on the *communicative* aspects of language learning [10], that is, the educational target is to improve communicational skills of students, in this case reading skills. Although the program is primarily developed as CALL application, it can also be regarded a *tool* for text comprehension, in that it assists people who know a bit of French but cannot read it quickly or reliably due to the presence of a number of unknown words in the text [5]. The potential practical use of Glosser-R$\underline{u}$G, therefore, is not restricted to educational tasks, but also includes supportive tasks, for example, providing employees of Dutch companies with a tool for reading French reports.

A further goal of the GLOSSER project, is to develop the application for a number of delivery platforms, such as the UNIX platform, the Windows-NT platform and the *World Wide Web*, thus enhancing the wide range of fields where applications developed within GLOSSER might be used. The fact that the focus of the project is on *support*, not drills, this line of applications typically does not require supervision, so that the potential scope is almost unrestricted. An early spin-off of Glosser-R$\underline{u}$G has been developed in Groningen, to study the feasibility of Glosser-like Web-applications. The result of this study is outlined in the WELLS project [1]. This application enables a user to exploit resources on the Web, such as dictionaries, thesauri etc., to support reading of a text in a foreign language. Although WELLS does not implement the complete functionality of Glosser-R$\underline{u}$G, the project clearly shows that the Web is, in a technical sense at least, a suitable medium for this line of CALL applications.

Although Glosser-R$\underline{u}$G is specifically designed for French texts, the architecture is modular, so that the application is easily adaptable to processing any language for which the informational sources used in the application are available. Moreover, a number of tools have been developed to enable the maintainer of the program to modify the informational sources the program uses, see for instance [4]. The program in principle organizes and integrates several previously developed linguistic tools and existing resources, in accordance with the goal of the project. The major informational resources that are incorporated are: an on-line dictionary provided by Van Dale Lexicografie [7], morphological analysis software provided by Rank Xerox [2], and examples of the use of words in especially collected text corpora. The program relies heavily on the morphological analysis software, which provides the link between the text that is being processed, and the other informational sources. Each word for which information is being sought is analyzed within its linguistic context (the sentence it occurs in) and disambiguated with respect to its base form, part-of-speech (POS) and the properties of the word that determine its appearance (or, inflections). The analysis thus provides the link between the possibly inflected form of the word as it appears in the text and the base form, or “citation form”, listed in dictionary entries [11]. This so called ‘lemmatization’ is also exploited for lookup of examples. All various forms of a word occurring in the text can be referred to by a single lemma, and can thus be indexed by this single form. This effectively guarantees a wide variety of inflections and lexical environments in the examples for a specific word, providing a rich context for vocabulary improvement and word comprehension. *Lemmatization* thus plays a dual role: it is informational in that it displays properties of words, and it is functional for processing, by providing the link between text on the one hand, and dictionary and examples on the other.

2 The Front-end of Glosser-R*u*G

The application can better be understood by decomposing it into functional elements. The individual elements then are: the front-end, that is, the user-interface, the dictionary, morphological analysis, example lookup, and the help-function.

The front-end of Glosser-R*u*G is constituted by the window-oriented user-interface, consisting of five main windows. Three windows are used to display the information Glosser-R*u*G provides on a word: a dictionary, morphological analysis and examples. The remaining two windows are used to display a read-only editor, and a hypertext-oriented help system. Figure 1 shows the complete screen the user gets to see when all modes of information are activated. (The help screen has been omitted.)

The left window in this figure is the read-only editor in which the text the user wishes to read can be displayed. There are three windows on the right. The top window gives dictionary information, the middle window displays the morphological analysis, and the bottom window provides examples of the selected word in other texts. The three windows displaying the informational sources and the help-window are *sticky*, that is, they cannot be moved or removed by the user. The editor window simply behaves according to the local window-manager; in general it is resizable, removable, etc. If the editor window is iconified, the sticky windows will stay on the screen. To iconify the whole application therefore, these windows have to be deactivated (that is, removed) by the program control, as explained below. Figure 2 shows the detailed appearance of the editor, when a text is displayed.

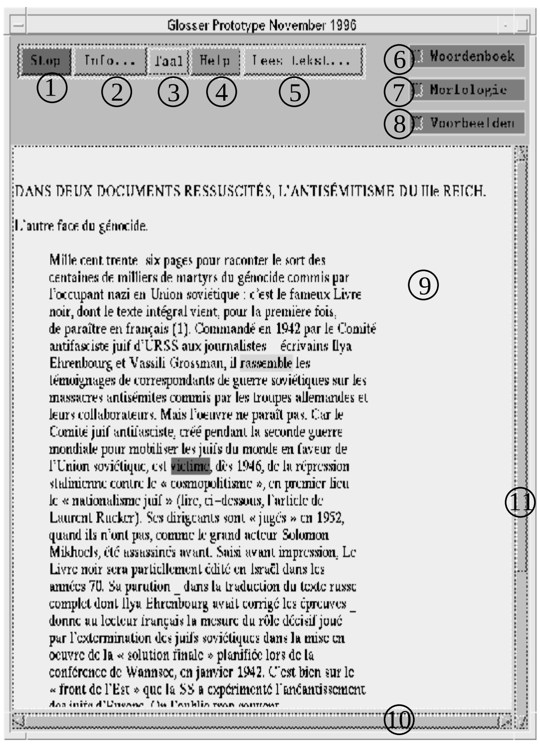

Figure 2: *The read-only editor. Numbers refer to specific functional parts of this window and will be explained below.*

The top of the editor window controls Glosser-R*u*G to a large extent. A horizontal menu displays five buttons. Button ① exits a session on Glosser-R*u*G. General information about Glosser-R*u*Gis given by ②, see section 2.1. The language switch ③ is not functional in this prototype, only sources for French-Dutch lookup are provided; user selections do not initiate any action of the program. On selection of Button ④, the hypertext-oriented help function is displayed with some general information on how to use help. The rightmost button of the menu, button ⑤, causes a file selection menu to be displayed, where the user can click on a file that is to be displayed in the text window ⑨. Selecting a directory instead of a file, causes the menu to display the contents of this directory,

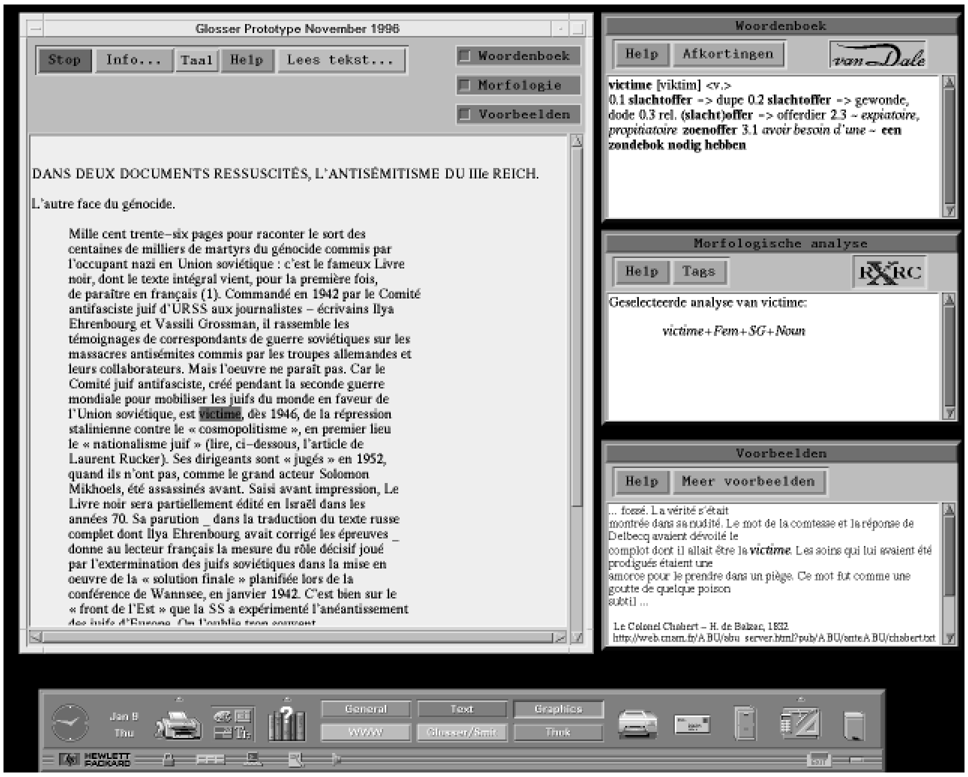

Figure 1: *Glosser-RuG as it is displayed on the screen. A help screen is also available, but not shown (see Figure 6).*

upon which a further selection can be made.

In the upper-right corner of the editor three on/off-switches control the specific sources that are to be used for a single lookup. ⑥ selects the dictionary, ⑦ selects the morphological analysis to be displayed (this analysis is conducted anyway, see section 3.1) and ⑧ opens the window for display of examples. The sources can be switched on and off at any stage during a session, so that a user can specify exactly what kind of information is necessary for a given word.

The last two numbers, ⑩ and ⑪ respectively, refer to the horizontal and vertical scrolling bars, used to display further parts of the text when it does not fit into the text window completely.

The information on morphological analysis is displayed in the window in figure 3. The user gets information on the part-of-speech that the specific word constitutes in the text, the base-form, or lexeme, of the word, and, possibly, a series of tags referring to properties of the word that cause specific inflections. These tags thus account for the specific appearance of the lexeme in the text. The analysis displayed in 3 for instance, shows that the underlying lexeme of *victime* as it appears in the text (see 2), is *victime* as well, that it is a noun and that the specific appearance of the surface form is determined by the fact that *victime* is singular and feminine. Two buttons specify further information on morphological analysis:

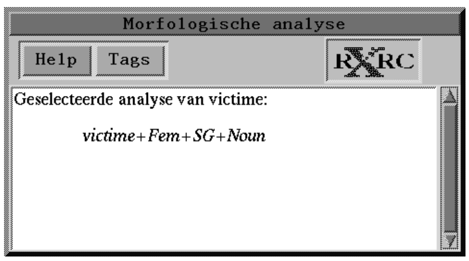

Figure 3: *Morphological analysis.*

Help provides specific information on the display of morphological analysis, and *Tags* shows a list of the tags used by Locolex, and their French translations. Both functions exploit the hypertext help-system of Glosser-R$\underline{u}$G, see section 2.1.

The window in Figure 4 displays a selection of the complete information available for an entry in the Van Dale dictionary. A number of mark-up tags has been omitted. The information that is displayed includes:

- the entry
- phonetic representation
- grammatical category
- translations and examples
- synonyms

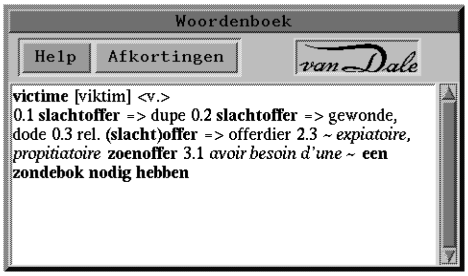

Figure 4: *The on-line dictionary.*

The dictionary window also provides context sensitive help under the *Help* button, and a list of abbreviations used in the dictionary entries, under *Afkortingen*. Both are displayed in the standard help environment.

Examples are displayed in the window shown in Figure 5. An example consists of a part of text in which the word that is submitted for lookup occurs, possibly in a different form. This word is highlighted by typesetting it in bold italics. The length of the text is fixed to a preset number of bytes left and right of the word, where incomplete words on the boundaries of the fragment will be omitted. If information on the source of the text is available, it will be displayed in a small font at the bottom of the display. The *Help* button provides information on this specific window; the button *Meer voorbeelden* selects the next example, if there is one.

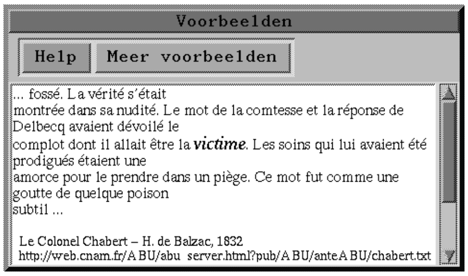

Figure 5: *Examples.*

2.1 Help

Glosser-R*u*G is supported by a hypertext oriented *help* function that provides general information on Glosser-R*u*G, and context sensitive help when called from specific environments, for example the dictionary window.

Figure 6: *The hypertext help-function.*

Figure 6 shows the window that is opened when help is started, in this case from the *Help* button on the main menu, that causes the window to display general information on how to use the function. Hyperlinks, that is, text fragments that are linked to other text, are displayed in red. A single mouse click on a hyperlink causes the related page to be displayed. Apart from this associative browsing, one can also recall the previous page by the *Terug* button, or go to the index of all available pages by using the *Index* button. The window is closed by clicking the *Stop* button.

Help provides information on the following topics:

- The use of the *help* function
- Glosser-R*u*G in general
- The editor window, and main control
- The *Dictionary* window

- Abbreviations used in the display of the dictionary
- The *Morphological Analysis* window
- The tags used by Locolex
- The *Examples* window
- Available language modules

3 Underlying Architecture

The next sections will present some of the underlying mechanisms of Glosser-R*u*G. A schematic outline of the communication between the elements that will be discussed and the front-end is shown in Figure 7. This figure represents the flow of information for a single search, given that all sources are switched on.

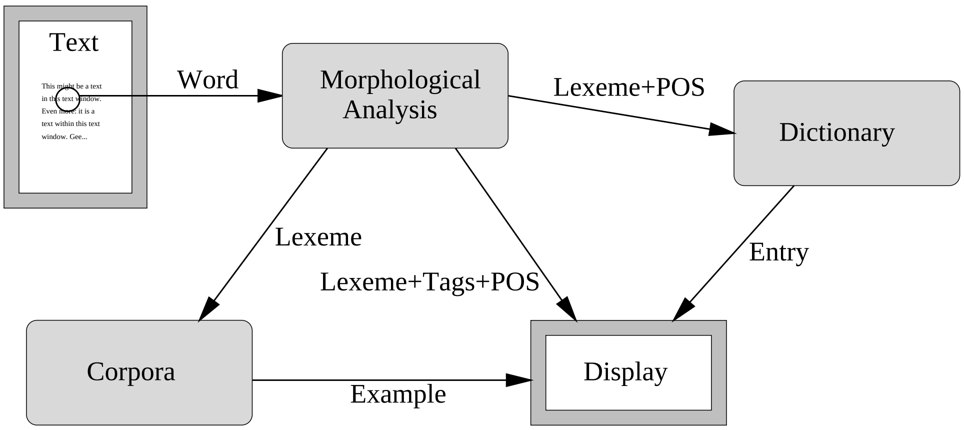

Figure 7: *The flow of information between functional modules of Glosser-RuG for a single search.*

After a word is selected for lookup, this word and the sentence it occurs in are submitted to Locolex for analysis. (A sentence is delimited by a period, a question mark or an exclamation mark for the algorithm; this over- and under specification will be improved upon for further versions of Glosser-R*u*G.) The lexeme thus found, will be displayed with tags and POS, but will also serve as input for the dictionary with the found POS, and for example search, where only the lexeme is used.

3.1 Morphological Analysis

Morphological analysis in general provides a framework for a wide variety of tasks. It is necessary for lemmatization, and extremely useful for part-of-speech (POS) disambiguation. Specific inflections offer valuable information on the possible syntactic categories for a word. One step further, lemmatization is a necessary prerequisite for dictionary lookup in most cases, since in general dictionaries do not contain inflected word forms as distinct entries. POS disambiguation is a useful tool to limit the ambiguity in deriving lemmata from inflected words, and also to limit the number of possible translations for words that occur with different syntactic roles, since every possible syntactic role for a word is a distinct entry in the dictionary. The French word *baiser* for instance, might be a transitive verb, where it means *to kiss*, or it might be a masculine noun, translated as *a kiss*. POS disambiguation clearly resolves this translation ambiguity. Glosser-R*u*G was fortunate to be able to incorporate *Locolex*, a state-of-the-art morphological analysis system provided by Rank Xerox [2]. Locolex accounts for both POS disambiguation and lemmatization. Moreover, the application provides a full analysis of a word regarding the inflections that cause its specific appearance. Here are some examples of the kind of analyses Locolex produces.

word	**analysis**	**lexeme tags**	**POS**	
vont	aller+IndP+PL+P3+FinV	aller	IndP+PL+P3	FinV
chats	chat+Masc+PL+Noun	chat	Masc+PL	Noun

Lemmatization is used both for dictionary lookup and for indexing examples in corpora. The former use is described earlier in this section, the latter will be further explained in section 3.3. Apart from lemmatization another feature of Locolex, POS tagging, is exploited for further processing. The POS tags are used for dictionary lookup in the cases where a word has more than one entry. (A word has a multiple entries *iff* it occurs in different syntactic roles.) In this case Glosser-R*u*G returns the entry for which the grammatical tag in the dictionary is consistent with the POS tag of the selected word. To be able to analyze individual words Locolex not only uses the specific word that is to be analyzed, but also the sentence the word occurs in. The sentence thus serves as linguistic context for determining the probabilities of specific POS analyses for a word. In cases where multiple analyses are regarded possible by the Locolex algorithm, the most probable analysis will be tagged. That is, all possible analyses will be returned to Glosser-R*u*G, but only the most probable analysis will be used for further processing.

3.2 Dictionary Lookup

The information on a word made available by a dictionary will undoubtedly be regarded the most important source by most users. A user study accordingly showed that the on-line dictionary is the most frequently used source of information [3]. Glosser-R*u*G was fortunate to be able to incorporate a French-Dutch dictionary by Van Dale Lexicografie [7], the premier Dutch publisher of dictionaries. The dictionary as it was made available consists of a number of tagged files. Each entry in the dictionary consists of a *searchterm* and a number of properties. To enable fast lookup of words, an index has been generated, where entries are <*searchterm, category, file, position*> quadruples. This (sorted) index can be accessed using binary search. For lookup, lexemes are used as generated by the morphological analysis, since inflected forms are not present as distinct entries in the dictionary. To disambiguate words that have multiple entries in the dictionary, due to the possible syntactic roles with which they can occur, the POS as found by Locolex is exploited. This ensures that the entry that is returned is syntactically consistent with the word as it occurs in the text. The table below shows some quantitative data on the dictionary:

Table 3.1

Entries	32017
Unique entries	31322
Translations	91690

Unique entries are those entries that only occur with one possible syntactic role. A word like *baiser* thus has two entries in the dictionary; one in it's grammatical role as *noun* (translation: a kiss), and one as a *verb* (translation: to kiss). POS analysis resolves those cases, although it is clear that these cases are very restricted in number. Still, the number of translations, even for a single entry, may be very large. Selecting the correct translation might prove to be tiresome for the user, in particular the user that is unfamiliar with reading French texts, for which Glosser-R*u*G is developed in the first place. A suggestion for resolving this problem is given in section 7.

3.3 Examples

An important feature of Glosser-R*u*G is that the application enables the user to view examples of words as they are used in other texts. The examples may occur in various forms, that is, inflections, by means of a lexeme based index (see 3.1). Also, the examples stem from a wide variety of different texts. This method provides the user with a large variety of lexical environments for a word, which has been shown an important factor in vocabulary learning [8], [6] (see also section 1). Ca. 5 Mb of text is used for these examples, accounting for a coverage of 7892 different lexemes [1] of the French vocabulary. This is still only ca. 30% of the entries in the dictionary. Future work will therefore include extension of the corpus, to approximate a coverage of 100% of both the dictionary entries and the 20.000 most frequently used words in French text.

The texts are indexed by determining the lexemes of the individual words using the same morphological analysis technology as described in section 3.1, and creating an index of N occurrences of each lemma thus found [4]. In this prototype N is set to 10, which is assumed a sufficient number

[1] Note that these are the lexemes as found after processing the text with Locolex, that is, the number of inflected forms in the corpus is much higher.

of examples for a specific word. This sorted index is accessible for binary search and consists of <*lexeme, filename, beginbyte, endbyte*> quadruples to mark the precise location of the word in the corpus. (The *endbyte*-marker is necessary for determining the word end for inflected forms, which might be shorter or longer than the lexeme in the entry.) The index thus provides a link between the lexemes and the possibly inflected forms, which enables the user to see examples not only of the specific word as it occurs in the text that is being read, but also other possible forms.

4 A Session with Glosser-R*u*G

To fully understand and appreciate the use of the application, this section steps through the various stages in lookup for a single word.

After the application has been started, the editor as shown in figure 2 appears. The only action that a user can invoke at this moment is an action on the main menu. At this stage, this means that a user can either exit the program, invoke help, or select a file to be read into the editor text frame. Upon selection of a file, it is displayed, and the user can now start a session.

First, the sources of information to be displayed have to be specified (although this is not a *necessary* step for the program code). It is assumed here that all sources will be switched on. Then the user has to select a word, for which information is sought. This is achieved by passing the mouse cursor over toward the text frame of the editor window, where a highlight marks the word currently under the mouse. A user can thus move the highlight to mark a specific word. A highlighted word is selected for lookup simply by clicking the right mouse button. After a delay of ca. 2 seconds, necessary mainly for the external call to Locolex (see section 6), the requested information is displayed in the appointed windows. Sometimes the user will have to scroll in the dictionary window, since entries contain more information than the window can display in the text frame. A further action might include looking up more examples for the same word, or getting help on the tags used for morphological analysis, or abbreviations used in the dictionary. This is the general procedure for lookup of a single word, which can be repeated for each unfamiliar word in the text. At any stage in the session the informational sources can be re-specified, that is, one could simply skip morphological analysis or any other source one likes for a specific word by closing the appropriate window.

5 New Features of this Prototype

Since the presentation of the first prototype of Glosser-R*u*G [5], a number of changes has been implemented, concerning both the algorithm and the appearance of the application. Some of those features were added as a result of a user study [3], where the practical performance of Glosser-R*u*G could be observed in a realistic user environment, and users could criticize the application and give suggestions for improvement. Other changes have been made, based on performance data and computational efficiency. This section provides an overview of major and minor changes, and the motivation for incorporating them.

5.1 Automatic Word-Highlighting

In the previous prototype selection of a word for lookup was achieved by clicking at one end of the word and then dragging the mouse-cursor over it with the left button held down. This method proved to be error-prone during the user study, partly due to inexperience of some users with the mouse. Also, numerous sessions on a laptop with a mouse/joystick embedded in the keyboard proved this method to be tiresome. The main cause for erroneous selections involves the issue of control. Using the method described above, where the user is in control, it is possible to select improper strings, like numbers, more than one word, etc. In some of these cases the application can transform an improper selection into a proper one, for instance by removing trailing spaces, punctuation, etc., in some cases an error message has to be displayed. This introduces some computational overhead: for every selection the user makes, the program has to trim off spaces or other non-alphabetical characters and check whether the selection is a proper selection for lookup, that is, whether it actually consists of a word. Also, bothering the user with error-messages, clearly effects the ease of using the application. Further, after a proper selection was made, it had to be submitted for a search explicitly, by clicking on a *submit*-button. This explicit action was regarded redundant by most users.

The whole selection-procedure therefore, has been drastically revised. The new prototype automatically selects words currently under the mouse and highlights this selection. Moving the mouse-cursor to a word therefore, is sufficient for making a selection. Since selection is controlled by the application, it is a predefined string such that any selection is always a proper selection. A single mouse-click starts a search action for the highlighted word. This method of course, is equally inefficient in a computational sense, since for every mouse position the word under the cursor has to be highlighted, thus it has to be checked whether it is a proper selection, but the important effect of this method is that user control is limited to starting a search action. It is considered of major importance that user invoked actions always lead to a positive result, not including error messages.

5.2 Indexed Dictionary Entries

A new method of dictionary access has been implemented in this prototype. Previously, the program code accounted for selection of the proper dictionary-file for a given word, while the file itself was accessed by a linear search routine. Although the dictionary files are rather short, this procedure is still very unsophisticated. Since the average file consists of ca. 4000 lines, the average number of lines that has to be read is ca. 2000. Binary search would already change this figure considerably, ca. 12 lines would have to be read for each search. However, one still has to specify the correct file for an entry. To simplify matters therefore, a single alphabetically sorted inverted index has been generated, for all entries in the dictionary. This index consists of <*Entry POS Filename Byteposition*> quadruples and can be accessed using binary search. Since the number of entries is 32017, the average (and largest) number of lines that has to be processed is $log_2 32017 \approx 15$. After the correct entry has been found in the index, the dictionary file is accessed directly at the precise location (*Byteposition*). This improvement is only 'visible' to the user by a decrease in processing time, see section 6. However, to some extent lookup speed is important for the functionality of CALL applications, and to the ease with which users read a text. Fast lookup guarantees that the entry is immediately displayed when the user needs the information, that is, at the moment the user is most actively concerned with that specific word.

5.3 Indexed Examples

Previous versions of Glosser-R$\underline{u}$G used the standard UNIX utility *grep* to extract examples of a given word from a collection of texts (although experiments with indexed corpora had already been conducted). There is a number of disadvantages to this approach. First, the result of the search is a number of examples in the same order as they are found. This implies that subsequent examples will often stem from the same text, therefore, are likely to be rather homogeneous with respect to their lexical environment. Second, this process is very inefficient with respect to processing time. Third, and probably most important, this method exploits string matching to find examples, which implies that other possible inflections of the selected word will not be included in the set of matches, which again amounts to a very homogeneous collection of examples.

To improve the example lookup, a single indexfile of the corpus used for examples was created, where entries were not strings (words) as they occurred in the corpus, but the 'lemmatized' forms, as found by Locolex. This means that the index maps lexemes onto all inflected forms present in the corpus. Since 10 examples per lexeme was regarded sufficient, a semi-random collection of references for each lexeme was sampled. This method thus amounts to a very heterogeneous set of examples. Further, the alphabetically ordered index offers a fast binary search access, such that <*example file beginbyte endbyte*>-quadruples can be found quickly. The file where the specific example is to be found can be accessed with *random access* directly at the given position, and the example retrieved, with a (developer-)predefined left and right context.

6 Temporal Performance

The data discussed in this section are concerned with computational performance of Glosser-R$\underline{u}$G. The educational qualities of the application will not be discussed in this report; for an account of these data see [3]. A comparison will be made to the previous prototype, to get an impression of the computational results of some of the extensions and changes. Data include processing times for: dictionary lookup, morphological analysis and example search. Processing times given are an average over 89 words from a single text. (Since there is no memory of previous word-lookup present in the program and its components, there is no bias whatsoever introduced by using one text.) Times in the table below are given in seconds.

Table 6.1

	new prototype	**old prototype**
Locolex	1.579	1.684
Dictionary	0.134	2.380
Examples	0.139	2.622
Total	1.852	6.686

These data show a clear difference between the two prototypes for the processes where changes have been made, namely the dictionary lookup procedure and the indexing of examples. The procedure of calling Locolex is virtually unchanged. Although Locolex clearly is the most time consuming part of Glosser-R*u*G, this algorithm is beyond the reach of the development: it is an external, compiled source. The specific use in the application, that is, an external call, does not effect Locolex's time consumption: stand-alone processing yields similar results.

7 Future Work

Future work for Glosser-R*u*Gwill include a number of improvements on existing methods, and some new features. Improvements include:

- Creating a two-stage index for examples, where the first index will consist of every unique lexeme, with a begin- and end position pointer to a second index. This second index then, will list all examples for the lexemes. Given an average of N examples for each lexeme, this will save $log_2 N$ steps in the index. (Although updating this type of index is more complicated, the advantage for on-line use is regarded as more important; updating does not have to be particularly fast.)

- Direct communication with Locolex, such that temporary files are no longer necessary. This might improve the speed of morphological analysis considerably. Much of the processing time used by Locolex is caused by the need to load some data files into the application. A constantly opened communication channel between Locolex and the other routines of Glosser-R*u*G offers a solution: data files would have to be loaded only once, at the start of the session.

- Incorporation of C-code (to replace TCL/TK code) in the application, to improve speed. This is especially suitable where lookup in the indexfiles is concerned. This issue extends to different *internal* representations of indexfiles, such that file access can be minimalized.

- The windows displaying the informational sources are safeguarded against deletion by the user, by removing the window-manager frame. However, unfortunately, this also implies that these windows are *sticky*, that is, immovable. The possibility to move these windows will be implemented for user convenience.

- The 'Meer voorbeelden' button on the examples window only enables forward browsing in the set of examples. A new version will enable the user to browse both forward and backward.

An extension that is beyond the scope of the current project, but would be a major improvement of the educational value of Glosser-R*u*G, is the addition of a system that could produce the correct pronunciation of a word. This can either be implemented by large a database or an algorithm creating on-line audio files. The most elegant solution is regarded to be on-line speech production, due to the productive nature of language, and the volume of a semi-complete speech database.

7.1 Major Extensions

Two specific extensions in functionality are regarded important for the improvement of Glosser-RuG for practical use. First, the use of bilingual texts for the examples would provide a user, especially the novice, with a better idea of how a given word is used in texts. Whether the translation will actually be displayed, can be up to the user. This extension has some practical consequences for the interface, as well as for the indexing of examples. The indexfile will have to consist of an entry with two begin-endbyte positions: one for the specific word in the corpus the lexeme refers to, and one for the translation. A one-to-one mapping from lexeme to translated word is rather complex. However, since examples consist of a text *fragment*, and bilingual corpora can be aligned, a mapping from lexeme to the complete aligned translation is an obvious and practical solution.

The second important issue is word sense disambiguation. Disambiguating a selected word with respect to the most probable translation or meaning is likely to improve the performance of the application, by selectively *decreasing* the amount of information made available to the user by the dictionary. Moreover, examples could be more specific. This will very likely result in a decrease in reading time and relieves the unexperienced reader of the task of selecting the correct translation. Also, providing more specific examples is expected to offer a better linguistic context for lexicon acquisition. It is thus likely that the user is able to look up far more words with this feature added, and probably achieve a better comprehension of the text. Moreover, the scientific question as to whether this technology is sufficiently sophisticated for this line of applications is an interesting issue in line with the general goals of the GLOSSER project.

A third possibility to improve practical use of Glosser-RuG would be to allow users to insert selected translations in the original text. Data on the searches conducted by single users on a single text showed that a number of words was looked up several times, which indicates that users do not remember each word that has been previously looked up. An obvious solution is to enable the user to make notes of translations found in the text. This idea has been confirmed by users' suggestions on improvements for Glosser-RuG, during a user study. Hence, a new prototype will enable the user to click on translations found in the dictionary, upon which the chosen translation is inserted in the text, directly after the original word. A simple mouse-click on a translation removes it if necessary. In the first case, in order to select, the highlight follows the mouse in the dictionary window, in the same way as for selection in the text that is being read. In the latter case the cursor changes shape, so as to show the nature of the action, that is, deleting the translation-note.

A further improvement will be to enable users to override a preferred analysis of Locolex, by displaying all possible analyses of morphological analysis that this program considered, also those that are rejected. [2] If this implies that the lexeme is different from the Locolex analysis, new examples and a new dictionary entry will be looked up automatically. This feature is obviously intended for the more advanced user.

8 Software Realization

Glosser-RuG is programmed in the script language TclTk (which is really the language *and* the interpreter), except for the morphological analysis. This function is an external application, embodied by Locolex, and written in C. The script language implementation makes the program very easy to debug and to alter. This is considered an advantage during the development stage, worth the price in computational speed and efficiency in comparison to compiled code. Moreover, since the underlying algorithms of Glosser-RuG are relatively uncomplicated, lookup procedures are quite fast anyway, as shown in section 6. TclTk is available for most commonly used platforms, like UNIX, Macintosh and Windows, which ensures easy portability of code. Sources and code for Glosser-RuG have been set up in a very modular way. The organisation of the complete structure is as follows:

[2] This feature was actually present in a previous version, but in combination with the user controlled cut-and-paste like selection. A new version will be fully program controlled, in the same *highlight-follows-mouse* fashion as selection in the text

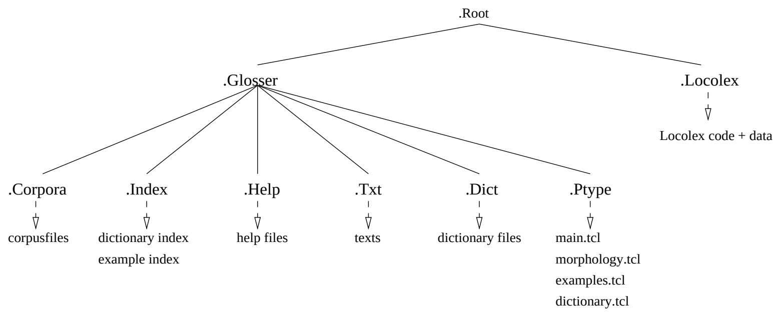

Figure 8: *The overall file structure of Glosser-RuG.*

9 Conclusions

Experience with development and users of Glosser-RuG has clearly shown that useful CALL applications based on state-of-the-art linguistic technology can be a realistic goal for short term projects. The availability of a number of tools, in a broad sense, that is, including scripting languages etc., makes development of the front-end of applications almost trivial and combining existing linguistic methods a matter of time. Present-day software and hardware support easy portability and distribution for most commonly used platforms, and for the World-Wide Web.

Acknowledgements

Development and research of Glosser-RuG is funded by the Copernicus project GLOSSER, grant number 343. I would like to express my gratitude to Prof. Dr. Ir. John Nerbonne, Alfa-Informatica, University of Groningen, for providing the ideal scientific environment for research, and personifying a constant source for consultation on all aspects of the project.

References

[1] A.Allodi D.A.Dokter and H.W.A.E.Kuipers. WELLS; Web Enhanced Language Learning. In *Language Teaching and Language Technology*, Groningen, The Netherlands, 1997.

[2] F.S.D. Bauer and A. Zaenen. Locolex: Translation rolls off your tongue. In *Proceedings of the conference of the ACH-ALLC'95*, Santa Barbara, USA, 1995.

[3] L.Schurcks-Grozeva D.A.Dokter, J.Nerbonne and P.Smit. Glosser-RuG; a User Study. In *Language Teaching and Language Technology*, Groningen, The Netherlands, 1997.

[4] D.A. Dokter. Indexing corpora for glosser. Techreport, Alfa-informatica, University of Groningen, Groningen, The Netherlands, 1997.

[5] J.Nerbonne and P.Smit. Glosser-RuG: in Support of Reading. In *COLING96*, Groningen, The Netherlands, 1996.

[6] G. Krantz. *Learning Vocabulary in a Foreign Language; A Study in Reading Strategies.* PhD thesis, University of Göteborg, Göteborg, Sweden, 1990.

[7] Van Dale Lexicografie. *Handwoordenboek Frans-Nederlands, 2^e druk.* VanDale Lexicografie b.v., Utrecht, 1993.

[8] J.-A. Mondria. *Vocabulaireverwerving in het vreemde-talenonderwijs; De effecten van context en raden op de retentie.* PhD thesis, University of Groningen, Groningen, The Netherlands, 1996.

[9] J. Nerbonne. Glosser- technical description. Project proposal, Alfa-informatica, University of Groningen, Groningen, The Netherlands, 1994.

[10] R.Last. Computers and language learning: Past, present - and future? In Chr.S.Butler, editor, *Computers and Written Texts*. Blackwell, Oxford, 1992.

[11] R.Sproat. *Morphology and Computation*. MIT Press, Cambridge, MA, USA, 1992.