Development of a Context-Sensitive Electronic Dictionary

Gábor Prószéky, Balázs Kis
MorphoLogic
Késmárki u. 8.
H-1118 Budapest
Hungary
{proszeky, kis}@morphologic.hu

Abstract

This paper introduces an electronic dictionary sensitive to the context of the input words or expressions. The dictionary program provides translations for any piece of text displayed on a computer screen without requiring user interaction. This functionality is provided by a three-layer process: (1) text acquisition from the screen, (2) morpho-syntactic analysis of the context of the selected word and (3) the dictionary lookup. Traditional dictionary entries need restructuring for this sort of usage. By dividing entries into smaller pieces and indexing them individually, the program is able to display a restricted set of information that is as relevant to the context as possible. For this purpose, we utilize automatic and semi-automatic XML tools for processing dictionary content. The construction of such an electronic dictionary involves natural language processing at almost every point of operation. Both the dictionary entries and the user input require linguistic analysis and intelligent pattern-matching techniques in order to identify multi-word expressions in the context of the input. By the time this paper is presented, the program incorporates more sophisticated language technology: multi-word phrases and sentences are recognized, and translation hints are offered in a linguistically intelligent way – by a parser/transformer module matching underspecified patterns of different degrees of abstraction.

Introduction

Most computer users encounter a large number of foreign language texts. They rarely need translations, only to read and understand the text. There are a large number of products facilitating translation, but still only a few projects attempt to assist users with the reading—or understanding—part. The foreign language comprehension assistant described in this paper is still not a translation program: it has evolved from a common electronic dictionary engine. Its features such as context analysis, however, provide a good support for foreign language text understanding.

Let us emphasize that developing such a tool takes more effort than adapting a bilingual dictionary, for many reasons. Bilingual dictionaries are traditionally composed in a format and structure suitable for assisting with translation. Foreign language comprehension, however, takes a different approach, and it is almost certain that any dictionary needs to be recompiled to some extent before it is incorporated in the comprehension assistant [Feldweg—Breidt 1996].

In this paper, we devise a context-sensitive electronic dictionary we usually label as a 'context-sensitive instant comprehension tool'. It suits any graphical computing environment: it reads text from anywhere on the computer screen, performs linguistic

analysis of the context of the input word, and uses an arbitrary number of dictionaries in the background. Finally, it displays context-dependent translations in a bubble—without requiring a single mouse click from the user, and leaving the screen contents intact.

In the following sections, we attempt to give an overview of the problem in general and the requirements of a comprehension tool. Then we describe the program's different phases of operation, with special attention to the linguistic elements and the dictionary development procedures employed throughout the application.

1 Sentence translation vs. word comprehension in context

It is clear that the linguistic process of understanding a passage of text is completely different from translating it. It is less clear, however, that it takes a different approach to implement a computational tool to assist the user with the task. A translator (either human or computational) has to analyse and transform every bit of the source text (and make her/his/its way through the ambiguities and unclear syntactic structures). Users attempting to understand an e-mail message—or any text that is displayed on a computer screen—need only hints about it, but they need it as quickly as possible, and without being disturbed, or even mislead, by ambiguities and other types of non-relevant information.

Translation support systems usually comprise of electronic dictionaries, less often aligners, translation memories etc. Such programs present a rather 'intrusive' user interface that draws the user's attention from the text to be understood. Document windows and dialog boxes are too disturbing for the user who needs 'in situ' information about a passage: he needs it exactly where the text appeared and he will not want to start another application, copy the text into it, and click some buttons to see translations. He will want assistance within the application he is currently working with.

Our comprehension assistant fully complies with this requirement: to display the context-sensitive translation of a word or a multi-word expression, all the user has to do is move the mouse pointer over the text in question. Context-sensitivity means that the program indicates if the word is part of a multi-word construction and will select the appropriate translation depending on the syntactic context if possible [Segond—Breidt 1996]. The translation itself is displayed in a bubble over the current screen contents, and disappears when the mouse is moved.

2 Requirements of a context-sensitive comprehension assistant

An instant comprehension assistant must work entirely alone in the sense that it cannot ask for user interaction: it cannot require the user to choose from a list of ambiguous linguistic analyses, and, at the same time, it should keep the proportion of semantic ambiguities as low as possible. Such applications can only rely on their own linguistic knowledge.

When the mouse pointer is left over a word for more than one second, it indicates that the user needs information about that word and its context. The boundaries of the context are not precisely specified: it could be the entire sentence (or even a larger passage) which includes the selected word, or—more often—a smaller context such as a multi-word expression around it. It is therefore the task of the program to determine the largest possible context, analyze it, and provide as much information on it as possible, based on the dictionaries behind the system. The minimum requirement is that the program should recognize all

obvious multi-word expressions and idioms, and provide appropriate translations. All possible forms of the multi-word expressions should be identified, even if word-forms are inflected or the word order is different from the basic form. This is the matter of the quality of the linguistic parsing components and the dictionaries. If no multi-word expressions are recognized in the context, the comprehension assistant should display a simple dictionary entry for the selected word only, listing all possible translations found in the active dictionaries. (See Figure 1 below.)

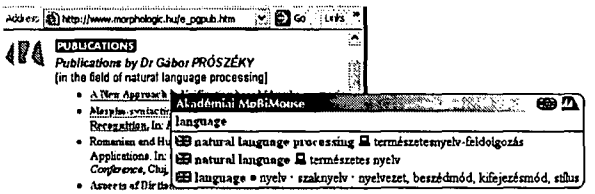


Figure 1: Dictionary information displayed by the comprehension tool, using an English-Hungarian dictionary

There is another implication of the fact that the comprehension assistant is not allowed to ask for user interaction. The program has to acquire pieces of text from the screen regardless of the application that displayed them without relying on user input, clipboard or file contents, or special application properties. As there is no direct access to the text, the program sees pieces of text as sequences of characters without formatting or other document-specific information, including the language of the source text. This requires implementing a language identification algorithm, too. It is clear that a useful comprehension assistant must be a rather special combination of different techniques, involving language technology in almost every bit of operation.

3 Phases of context-sensitive instant comprehension

The program continuously watches the motion of the mouse pointer and acquires the text from the screen position where the mouse pointer stops. This phase is called *text acquisition.* Current implementations rely on operating system resources to acquire text displayed on the screen. Our implementation performs an OCR-like procedure on the screen contents. This phase acquires one or more lines of text around the cursor position, and this additional information is later analyzed by the second phase. (See Figure 2 for demonstration.)

Linguistic analysis, the next step, is supposed to identify the word that was pointed at, and perform a morpho-syntactic analysis of its context to determine the headwords to look up in the dictionaries. Linguistic analysis consists of several steps essential for proper dictionary lookup, because there is no initial information about the text other than the text itself—with a single word highlighted indicating the position of the mouse pointer and thus the initial point of the analysis. It is very important to note that these initial data are results of an OCR-like process whose errors require correction during subsequent linguistic analysis.[1]

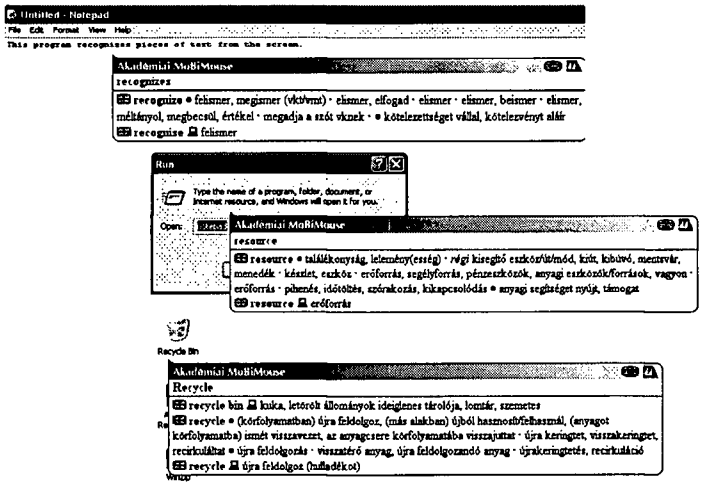


Figure 2: Different text recognition situations

The linguistic analyzer module performs morpho-syntactic analysis for the selected word in context—by means of the HUMOR engine [Prószéky—Kis 1999]. At this point, morphological analysis has three main purposes:

(a) linguistic stemming for accurate dictionary lookups (see Figure 3),

(b) spelling correction and

(c) preparation of shallow parsing of the context to identify candidates for multi-word expressions.

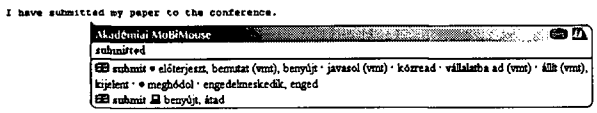


Figure 3: Stemming in action

If linguistic analysis fails to recognize any multi-word expressions, words from the context are still passed on to the dictionary lookup phase as the dictionaries may contain idiomatic phrases that cannot be recognized on a linguistic basis.

The *dictionary lookup* phase receives lexical stems of the selected words and words in the context, which are then matched against all dictionaries. We utilize an intelligent dictionary engine capable of handling multiple dictionaries at the same time [Prószéky 1998]. Dictionaries are often adapted to comprehension needs by filtering out non-relevant information [Feldweg—Breidt 1996]. Multi-word lexemes are found by forming sequences of the received lexical stems, and querying the dictionaries for the sequences. However, the dictionary engine is capable of finding all multi-word lexemes which include one single word with a single lookup. In some cases, this could be a rather lengthy list which must be filtered by using the other words in the context. More precisely, (translations for) multi-word

expressions will be displayed if and only if they include some significant words of the context (and do not contain other significant words). By 'significant word', we mean that there are also 'non-significant' words (or stop-words) that are skipped when forming a query expression for the dictionary engine.

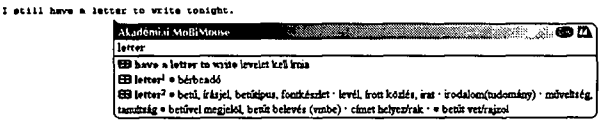


Figure 4: The comprehension assistant identifying a multi-word expression and finding it in the dictionaries.

The output of the comprehension assistant is shown in a bubble-shaped pop-up window on the screen that disappears if the user moves the mouse cursor again. Figure 5 describes the layout of a dictionary entry bubble displayed by the comprehension assistant.

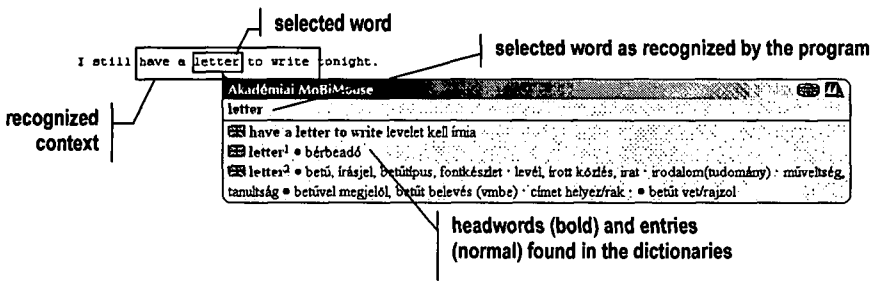


Figure 5: Parts of a dictionary entry as displayed by the comprehension assistant in the bubble.

4 Aspects of dictionary development

Dictionaries in our system are represented as lexical databases where the structure of each dictionary is strictly preserved. This is achieved through using XML as the single dictionary format. Dictionaries are either originally written in XML or transformed from a printed or another electronic format by means of automatic and semi-automatic tools.

In Section 1, we mentioned that any dictionary must be restructured in order to be used with a comprehension assistant. This restructuring step is largely unavoidable as there are no dictionaries compiled directly for this purpose. In the examples of the present paper, we use the newest and largest English-Hungarian dictionary that was compiled using XML techniques—but primarily for the purpose of a printed dictionary. The XML format, however, facilitated the inclusion of the dictionary in 'traditional' stand-alone dictionary programs.

In the following description, we summarize the most important aspects of restructuring dictionaries to suit comprehension assistance.

As a first approach towards the intelligent treatment of multi-word expressions, we needed to recognize that the sample phrases and sentences in a large dictionary provide enough multi-word expressions for the program to be useful. This is very important for providing translations—as the translations are there in the dictionary—, because as long as we have nothing more than a dictionary program, the only way to construct translations is to read them from dictionary databases and display them as they are stored.

As a basic rule, it is also clear that we cannot display entire dictionary entries. The main reason for this is *not* that they tend to be too lengthy (and could easily cover the entire screen) in large dictionaries—although this is also an aspect that has to be taken into account. The real requirement is that the reply of the program must be relevant to the context.

For this reason, original dictionary entries must be split up into smaller parts: all examples and otherwise included multi-word expressions must be treated as separate dictionary entries. If the original dictionary is properly tagged into XML, the recompilation process is largely a mechanic conversion.

The above step is the most important in restructuring entire dictionaries. Thus all multi-word expressions in the dictionary are treated by the program as headwords. This is important because headwords are directly indexed and can be found in the database by a single lookup operation.

The reply of the program is rarely a single dictionary entry. As a general rule, the comprehension assistant finds all suitable entries for the context of the selected word, and displays them in the same bubble. The bubble in Figure 6 shows a reply composed of three separate dictionary entries, all relevant to the context of the selected word.

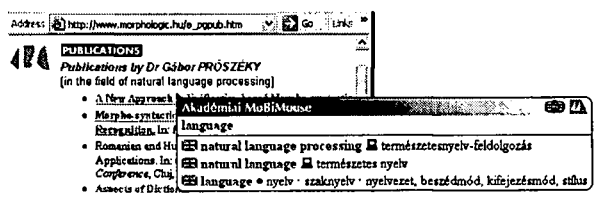
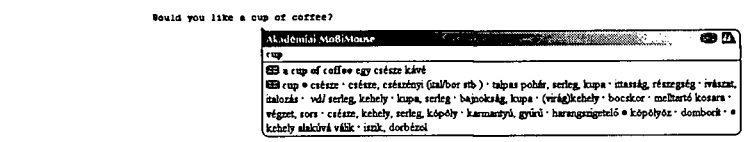


Figure 6: The comprehension assistant composes the reply from separate dictionary entries, displaying only those relevant to the context

It is also very important to index each multi-word expression word by word as they must be found by any of their words. The text acquisition module returns a whole line of text with one single word marked: it is, in a non-linguistic sense, in the 'focus' of the user query. The marked word can be any one of a multi-word expression. Thus the expression 'a cup of coffee' is found by pointing at either 'cup' or 'coffee', as shown in Figure 7:



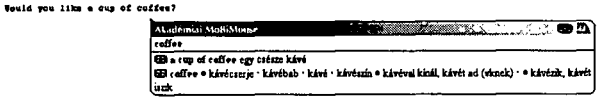


Figure 7: Finding the same multi-word expression by pointing at different words.

5 Supporting dictionary development through users of comprehension assistants

Although we thoroughly surveyed future users when starting the project, it has been clear from the beginning that existing dictionaries will be far from perfect at any stage of the project life cycle. Therefore, dictionaries are continuously reviewed and updated. The speciality of this process is that it is built largely on user feedback. From the aspect of dictionary development (and even linguistic research), the comprehension assistant is an ideal source because it reaches a potentially large number of users (since it is not a special application but a utility that has its place in every computing environment). Based on this insight, we have implemented an instant feedback feature, which comprises of two processes:

1. Logging: the comprehension assistant continuously logs words and multi-word expressions it was unable to analyze or failed to find in the dictionaries.
2. Contacting the developers: the program automatically sends e-mails containing the current logs to the developer lab.[2]

Logs are thus gathered and analysed by further computational tools in the development lab. Having been filtered to exclude obvious noise entries, the list is then sent to lexicographers for further analysis. This process effectively reveals errors and deficiencies in the dictionaries, and, at the same time, it helps defining directions of further improvements.

6 Comparison to other existing systems

There are two categories where our context-sensitive comprehension assistant tool—its brand name is MoBiMouse—can be compared to other systems: functionality and linguistic accuracy. There are only a few pop-up dictionaries on the market: some of the best known ones are Babylon, WordPoint or Langenscheidt's Pop-up Dictionary, but none of them have as many language technology features as MoBiMouse. There are some 'glossing' programs in research laboratories [Feldweg—Breidt 1996; Poznanski et al. 1998] that access dictionaries with a context-sensitive lookup procedure. However, they present the information to the user through their own graphical interface: none of them are accessible from any running application—which is quite essential to MoBiMouse. The above systems do not have access to more than one dictionary at the same time, unlike MoBiMouse. On the other side, the treatment of multi-word units in the IDAREX formalism [Segond–Breidt 1996] is more sophisticated than in MoBiMouse (although we outline a new approach of abstraction in section 7). Another project with instant understanding is GLOSSER, whose prototype [Nerbonne et al. 1997] also carries out the morphological analysis of the sentence in which the selected word occurs. In GLOSSER—unlike in MoBiMouse—there is a stochastic disambiguation step but everything is shown in a separate window.

The character identification techniques applied in our comprehension assistant are independent of both the language and the writing system: it is rather different from all known applications that work with English characters only. There is a version that recognizes complex scripts such as Japanese characters—which enables users to read and understand texts that they cannot even type in.

Other pop-up dictionary applications start by pressing a button or clicking the mouse. MoBiMouse is the only known application that starts without clicking; therefore it can be used to acquire any text from the screen without affecting other running applications.

The speed of the text acquisition module is 1000 characters/s, stemming is performed at 0.002 s/word-form, an average dictionary lookup takes 0.02 s.

MoBiMouse, unlike applications like Babylon, can be used in both language directions of the dictionary due to its writing independence and linguistic components for many languages.

7 Towards greater linguistic sophistication

In a separate project, we have developed a parser module that is capable of recognizing abstract linguistic patterns. The basic operation of this parser—named HumorESK[3]—is matching an underspecified linguistic pattern against a pattern dictionary, and creating a single abstract symbol from it [Prószéky 1996]. In turn, this abstract symbol can be part of other underspecified linguistic patterns, and thus participate in subsequent dictionary lookups. Parsing an entire sentence can take several hundreds or thousands of subsequent pattern matching operations. Still, the average parsing time for a sentence is less than 20 ms on an average PC[4].

The most important feature of this parser that makes it suitable for comprehension assistance is the capability of extracting recognizable elements from largely unrecognized structures. The parser does not expect to be able to parse the entire input; it can nevertheless mark and reveal all recognized structures—noun phrases, verbal phrases, and, in special cases, entire sentences.

The most important feature of the patterns matched by the HumorESK parser is that their elements are at different degrees of lexical specification. Let us consider the following example:

NN = NP(lex="bottle") + PREP(lex="of") + NP

This pattern matches all sequences where there is the noun phrase 'bottle', followed by the preposition 'of' with an arbitrary noun phrase at the end. Note that the pattern as shown here is simplified to a great extent, as there are several other properties assigned to each symbol. In principle, the abstract patterns implement a feature-based grammar where property matching and value assignment are parts of the pattern matching process.

For such patterns, however, there are no ready-made translations we can read from a database and display as they are stored there. There must be a formal—and partly algorithmic—way of constructing translations for abstract patterns. The HumorESK project is part of a larger one called MetaMorpho where each recognition pattern (see the above example) can be assigned one or more transformation patterns. Thus the parser is not only

capable of marking structures and providing their analysis trees, but in principle, it can transform the source text into any other formalism—where a translation in a different language is one of the possibilities. Here is an example of the transformation pattern (simplified the same way as the previous example):

EN.NN = NP#1(lex="bottle") + PREP(lex="of") + NP
HU.NN = NP#1[lex="üveg"] + NP

The transformation pattern is shown in the second line. This pattern controls the translation of the source phrase into Hungarian.

The above mechanism significantly increases the linguistic sophistication of a comprehension assistant as it is capable of recognizing and translating multi-word expressions not included in the dictionaries.

8 Conclusion

The context-sensitive instant comprehension assistant named MoBiMouse offers translations for words and expressions on the computer screen. The program is activated by moving the mouse pointer over a particular word: the translation is displayed in a bubble. If the mouse is moved away, the translation promptly disappears, so the user's attention is not distracted by yet another program requiring a whole window with all its features.

MoBiMouse has been written in C++ and can be used anywhere in a Windows (95/98/NT/ME/2000/XP) environment: in any application, on the desktop, in the help windows or even in menus. It recognizes all texts written in all installed fonts. It is especially useful when browsing the web: here it is the most probable that users encounter texts in foreign languages. MoBiMouse helps reading these texts fast and without interrupts and properly recognizes texts where there is a colourful (or patterned) background—so it can acquire pieces of text that our eyes are unable to read.

MoBiMouse performs extensive linguistic analysis of the selected word and its context, and provides intelligent replies for multi-word expressions where all translations are relevant to the context.

Acknowledgements

We would like to thank András Földes, the leading developer of MoBiMouse, who began working on this project as early as in 1996, and has completed the first working prototype in 1998. Since then, he keeps adding new features promptly and efficiently. This made it possible that all features of the program described in this paper are fully operational.

The examples shown in this paper are taken from a special version called Akadémiai MoBiMouse (Academic MoBiMouse) that uses the adapted version of the largest and newest English-Hungarian dictionary published by Akadémiai Kiadó (Academic Publishing House), Budapest, in 2000. The adapted version contains over 400,000 entries. We would like to thank the publisher for their co-operation.

Endnotes

[1] According to our experience, however, recognition errors occur quite rarely because there is a closed set of shapes (glyphs in the currently installed system fonts) that may occur in any text displayed by applications (except for pieces of text within bitmap images). Recognition errors are usually results of applications using non-standard techniques (e.g. dynamically altering character spacing) to display text.

[2] This requires permission from the user which the program asks for during installation.

[3] The name stands for High-speed Unification Morphology Enhanced with Syntactic Knowledge. The first prototype was an enhancement of the HUMOR morphological analyzer (mentioned earlier in the paper). Now HumorESK is an entirely different project with a new parsing technology.

[4] An Intel Pentium III-based PC at a minimum of 600 MHz.

References

[Feldweg—Breidt 1996] Feldweg, H. and E. Breidt, 1996. COMPASS – An Intelligent Dictionary System for Reading Text in a Foreign Language. *Papers in Computational Lexicography (COMPLEX 96),* Linguistics Institute, HAS, Budapest, pp. 53–62.

[Nerbonne et al. 1997] Nerbonne, L. Karttunen, E. Paskaleva, G. Pröszéky and T. Roosmaa, 1997. Reading More into Foreign Languages. *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP–97),* Washington, pp. 135–138.

[Poznanski et al. 1998] Poznanski, V., P. Whitelock, J. Udens and S. Corley, 1998. Practical Glossing by Prioritised Tiling. *Proceedings of the COLING-98,* Montreal, pp. 1060–1066.

[Prószéky 1996] Prószéky, G., 1996. Morphological Analyzer as Syntactic Parser. *Proceedings of the COLING-96,* Copenhagen, pp. 1123–1126.

[Prószéky 1998] Prószéky, G., 1998. Intelligent Multi-Dictionary Environment. *Proceedings of the COLING-98,* Montreal, pp. 1067–1071.

[Prószéky—Kis 1999] Prószéky, G. and B. Kis, 1999. A Unification-based Approach to Morpho-Syntactic Parsing of Agglutinative and Other (Hughly) Inflectional Languages. *Proceedings of the37th Annual Meeting of ACL,* College Park, pp. 261–268.

[Segond—Breidt 1996] Segond, F. and E. Breidt, 1996. IDAREX: description formelle des expression à mots multiples en français et en allemand. In: A. Clas, Ph. Thoiron and H. Béjoint (eds.) *Lexicomatique et dictionnairiques,* Montreal, Aupelf-Uref.