Natural Language Engineering **2** (3): 211–228

*Prototype of a second language writing tool for French speakers writing in English**

ETIENNE CORNU, NATALIE KÜBLER†, FRANCK BODMER‡, FRANÇOIS GROSJEAN, LYSIANE GROSJEAN, NICOLAS LÉWY, CORNELIA TSCHICHOLD§ and CORINNE TSCHUMI

Language and Speech Processing Laboratory, University of Neuchâtel, Switzerland

(*Received 21 May 1996; revised: 20 November 1996*)

Abstract

Language tools that help people with their writing are now usually included in today's word processors. Although these various tools provide increasing support to native speakers of a language, they are much less useful to non-native speakers who are writing in their second language (e.g. French speakers writing in English). Real errors may go undetected and potential errors or non-errors that are flagged by the system may be taken to be genuine errors by the non-native speaker. In this paper, we present the prototype of an English writing tool which is aimed at helping speakers of French write in English. We first discuss the kind of problems non-native speakers have when writing in a second language. We then explain how we collected a corpus of errors which we used to build a typology of errors needed in the various stages of the project. This is followed by an overview of the prototype which contains a number of writing aids (dictionaries, on-line grammar helps, verb conjugator, etc.) and two checking tools: a problem word highlighter which lists all the potentially difficult words that cannot be dealt with correctly by the system (false friends, confusions, etc.) and a grammar checker which detects and corrects morphological and syntactic errors. We describe in detail the automata formalism we use to extract linguistic information, test syntactic environments and detect and correct errors. Finally, we present a first evaluation of the correction capacity of our grammar checker as compared to that of commercially available systems.

Most of today's English word processing programs include a set of tools to help people with their writing. In addition to a thesaurus and a spell checker, they increasingly provide a grammar checker that identifies non-standard style and a number of grammatical errors. It is generally accepted that these tools provide good

* The research presented here was part of a three-year project funded by the Swiss Committee for the Encouragement of Scientific Research (CERS/KWF 2054.2). The authors would like to thank Jacqueline Gremaud-Brandhorst, Catherine Liechti and Alain Matthey for their help during the project. Requests for reprints should be sent to: Dr. François Grosjean, Laboratoire de traitement du langage et de la parole, Université de Neuchâtel, Avenue du Premier-Mars 26, 2000 Neuchâtel, Switzerland. E-mail: francois.grosjean@lettres.unine.ch

† Now at CERIL, University of Paris VII and at the Laboratoire de linguistique informatique, University of Paris XIII, France.

‡ Now at the Institut für deutsche Sprache, Mannheim, Germany.

§ Also at the English Seminar, University of Basel, Switzerland.

support to native speakers of English but it is not clear that they are as useful to non-native speakers. As many years of research in applied linguistics have shown, non-native speakers make errors that are quite often different from those of native speakers and from those of speakers of other languages. This is due to the fact that many of their errors are directly linked to their first language. Thus, for example, when speakers of French write in English, their errors will often be different from those of native speakers of English or, for that matter, from those of native speakers of Spanish or German. Current monolingual grammar checkers do not take these differences into account and hence are not as helpful as they could be to non-native speakers.

It was with this problem in mind that we developed a prototype of a second language writing tool aimed specifically at helping native speakers of French when writing in English. In order to meet all the writing and checking needs of these users, our prototype contains writing aids (dictionaries, an on-line grammar, a verb conjugator, etc.) and two checking tools: a problem word highlighter which lists all the potentially difficult words in the text (false friends, confusions, etc.) and a grammar checker which detects and corrects morphological and syntactic errors. (For other projects that have dealt with second language grammar checking over the last few years, see Barchan *et al.*, 1986; Yazdani, 1993; Granger and Meunier, 1994; and Brehony and Ryan, 1994, among others).

In this paper, we first present the problems non-native speakers have when writing in a second language. We then explain how we collected a corpus of errors which we used to build a typology of errors needed for the various stages of the project. This is followed by an overview of the prototype and of its writing and checking tools. Next we describe in detail the grammar checker: the linguistic information needed, the automata formalism implemented, and the detection, correction and interaction procedures used. We end with a first evaluation of the grammar checker as compared to that of commercially available systems.

1 Writing in a second language

As research in applied linguistics has shown, non-native speakers do not make the same kinds of errors as native speakers. The competence they have in their second language is influenced by their two languages, the first and the second (Corder, 1967; Selinker, 1972, 1992). The errors they make when speaking or writing their second language are of two types – interlanguage errors and intralanguage errors. Interlanguage errors, also called interferences, are due to the first language influencing the second language. For example, a French person might write, '*Sam steals money to Max', based on the French structure, 'Sam vole de l'argent à Max', where 'à' is translated with 'to' instead of 'from'. (Note that 'to' is the most frequent translation of French 'à'). Interlanguage errors can be found at all levels of linguistic analysis: orthographic (e.g. '*adress' based on French 'adresse'), lexical (e.g. '*the needles of the clock' based on French, 'les aiguilles d'une montre'), syntactic (e.g. '*he asked to his friend' based on, 'il a demandé à son ami'), etc. (see Grosjean, 1982, for an overview).

Intralanguage errors, on the other hand, are characterized mainly by over-generalizations or false generalizations which can also be made by native speakers of the language (e.g. children, adults under stress, etc.). They occur when a rule is applied incorrectly as, for example, when verbs with an irregular past tense are given the regular '-ed' form (e.g. *gived, *runned, etc.). Other types of intralanguage errors, more specific to non-native speakers this time, involve simplification such as dropping pluralization and tense markers, omitting function words, simplifying the syntax, dropping the auxiliaries, etc. Yet other errors involve hypercorrection and the avoidance of words, expressions and structures that may be too difficult for the non-native speaker.

Given the type of errors made by non-native speakers, it is no surprise that current monolingual grammar checkers are not really adequate. Those that use a complete parse approach have problems as errors not taken into account in the analysis lead to a parsing failure, and thereby to the frequent inability of the system to identify where the problem lies and what it is. For grammar checkers that use a less demanding approach, the error might be detected but it may not be identified properly and hence ways of correcting the error may not be suggested. Leaving non-native speakers to identify and correct an error that has been detected is not a good solution as they do not have the kind of language intuition that native speakers usually bring to a correction task. In addition, simply flagging potential errors that pertain to levels of analysis beyond the scope of the grammar checker (e.g. semantic errors) is inappropriate as, once again, non-native speakers do not have the competence needed in their second language to make the appropriate decision. Based on a potential error message, they might modify a word or a sentence that is not in fact wrong and hence make things worse. Finally, it is important that when writing in a second language, non-native speakers have available to them appropriate aids and help messages, in their native language preferably, so as to understand the problem fully.

In summary, grammar checkers for non-native speakers (also called bilingual grammar checkers) have an important role to play, all the more so if one considers the amount of writing that takes place today in people's second languages (for example, in English as a second language).

2 Error typology

Before starting to build our prototype, we undertook an in-depth analysis of the errors made by native speakers of French when writing in English. To do this, we identified four possible sources of information: teachers of English who are familiar with the errors French speakers make, published lists of errors made by learners of English, comparative analyses of French and English structures, and texts written in English by French native speakers. Although each source by itself was not fully sufficient for our needs (and sometimes caused problems as published lists of errors are not exhaustive, teachers' intuitions are sometimes different, etc.), we found that a combination of all four sources was the best solution. We paid particular attention to comparative analyses (see, for example, Vinay and Darbelnay, 1977; Guillemin-

Flescher, 1981; Kübler, 1992, 1995) and to texts written by French speakers. We collected high school and business school written exams and essays (27,000 words in all) and had them corrected by three native speakers of English (all ESL teachers). This produced 2862 errors to which we added those collected from published lists of errors and from comparative analyses.

Using these various sources of information, we built a typology of errors which we divided into seven sections. The first section concerned graphical form and contained errors caused by spelling (e.g. *appartement, *marchant, *developped), phonology (e.g. *catastrophy, *distroyed) and morphology (e.g. *certains childrens). Also included in this section were mixtures of British and American spelling (e.g. *honor and labour). The next four sections dealt with parts of speech: adjectives, adverbs, nouns and verbs. For each part of speech, errors were subdivided into those that pertained to morphology, tense, structure, the lexicon, etc. For example, the errors that concerned adjectives were divided up as follows: morphological errors (e.g. *He was the most small in the group), lexical confusions due to syntax (e.g. *It has a religion effect) or to semantics (e.g. *He has a bigger allowance and it's not just), nonce borrowings (e.g. *This phenomenon remains inexplicated) and syntactic errors (e.g. *In the few last years). The sixth section dealt with groups of words, such as compounds, idioms or support verbs followed by predicate nouns, many of which are translated literally into English (e.g. *I have cold) or not used correctly (*Is it really twice heavier?). Finally, the seventh section concerned errors that pertain to the sentence as a whole including negations (*She came not …), reference pronouns (*If the firm calls, tell her to …), word order (*He asked me when are we leaving …), agreement (*There is people who think …), subordinate clauses (*He was sent abroad for learning Spanish …), etc.

Our typology proved to be very useful throughout the project. We used it, among other things, to decide which errors to include in the various parts of the prototype: the writing aids, the problem word highlighter and the grammar checker. A number of factors were taken into account when deciding whether to cover an error, especially in the grammar checker. Among these were the frequency of the error (obtained from the corpus), its impact on comprehension and, above all, the computational complexity involved in detecting and correcting it.

3 The prototype

When designing our prototype, we tried to keep in mind a number of guiding principles. We wanted a tool that could be of help to non-native speakers while they were in the process of writing their text and during the correction stage once it was done. Thus, we decided that the prototype should contain both writing aids and checking devices. We also wanted a tool geared in every way to non-native speakers (in our case, French speakers writing in English). As we have seen, their writing and checking needs are quite different from those of native speakers and their feeling of security, or rather insecurity, concerning their written language is also quite different. We therefore wanted our prototype to be 'knowledgeable' about second language writing and its problems, and to be as helpful and as reassuring as possible. Other

guiding principles we gave ourselves were: to detect and correct as many errors as possible, to concentrate on the most frequent errors if at all possible, to limit overflagging (false alarms) to a minimum (see the reasons given in section 1), to have the system interact with the user whenever necessary (but not too often), to make use of up-to-date technology, and finally, to use linguistic databases that could be easily updated.

The prototype includes a set of writing aids, a problem word highlighter and a grammar checker. The prototype runs under Microsoft Windows and has an interface developed with Visual Basic. It is composed of a number of modules that were developed independently and its code is completely portable between the Macintosh and the PC. At the top level, the prototype provides an editing window where files can be opened, saved and closed and where text can be cut, copied and pasted. A number of writing aids can be accessed directly from the editing window through simple menu selection. They are:

(a) An English dictionary and a French-English bilingual dictionary (simulated at this stage).
(b) A verb conjugator which allows the user to enter a verb and obtain its forms in all persons and tenses.
(c) A set expression translator which accepts French set expressions and which produces their counterpart in English (e.g. for 'tenir compte de X' the translator proposes 'to take X into account'). These expressions cannot be translated literally into the other language and yet they are used quite often, hence the need for this kind of writing aid.
(d) An on-line grammar designed specifically for French speakers. It gives advice in French on specific problems that they have when they write in English, e.g. the tense system, prepositions, the use of the infinitive and the gerund, 'since' and 'for', word order, indirect speech, and so on. Each grammatical topic is explained in simple, non-technical language and examples are given. This aid is accessible both during editing and during the correction phase.
(e) Finally, a list of difficult words which includes false friends, confusions, irregular verbs and borrowings. For example, users are told that the adjective 'satisfied' is followed by the preposition 'with' and not 'of', and the verb 'to disobey' is transitive in English whereas it is followed by a preposition in French ('désobéir à'). Each word's difficulties are explained and examples of correct use are given.

The problem word highlighter, one of the two checking tools in the prototype, is used to show all the potential lexical errors in the text. When examining the errors in our typology, we quickly realized, like others have done before us (Catt, 1988; Thurmair, 1990; Payette and Hirst, 1992), that some errors could not be detected automatically given the current state of natural language processing. These include false friends (e.g. using 'library' instead of 'bookstore' based on French 'librairie'), confusions (e.g. French 'partie' can be translated as 'part' or 'game'), foreign loans (e.g. the word 'training' used instead of 'tracksuit' because it originally comes from English and means tracksuit in French), etc. These lexical difficulties normally need

semantic and pragmatic analyses to be dealt with appropriately. Clearly, most grammar checkers cannot do this, and they therefore fall back on flagging these errors without being able to determine whether they are used erroneously in the text. Most often this approach results in heavy overflagging. This is a real problem when one is dealing with non-native speakers who, on seeing such a message, may think they have made an error when this may not be the case. To avoid this problem, we decided to deal with potential lexical errors separately. Thus, users can activate this highlighter whenever they wish to and they have all the problem words in the text appear on the left-hand side of the screen. When they click on any one of these words, a text window appears which gives them information on the various uses of this word. Such an approach has two advantages. First, it is the user who decides on the words that need to be explained, thereby saving a lot of time. Second, problems that cannot be processed automatically can be handled here, thus reducing the false alarm rate of the grammar checker.

The second checking tool is the grammar checker. It can be run during editing or afterwards and works sentence by sentence. When it identifies an error, a window is displayed which contains two main text fields: the section of the text where the error is located, and an explanation of the error along with suggestions for correcting it. A number of options are then offered to the user: correct the error manually by editing the text, obtain more information on the grammatical aspects of the error, select a word to insert from a list of words proposed, click on a button to execute the action suggested by the grammar checker (insert, replace, delete or permute), or disregard the error and continue. The user can also open the grammar checker's option screen and activate or deactivate the error categories to be used. Below we describe in detail how grammar checking is done.

4 Grammar checking

The grammar checker uses a set of rules represented as a variant of finite-state automata (Winograd, 1983; Allen, 1987; Silberztein, 1993) which seek specific sequences of elements in the input text. The automata consist in lists of conditions that apply to adjacent words or phrases. They are initialized by an anchoring arc which is inside the automaton (and not normally on its left-hand side). The anchoring takes place when a word matches the conditions assigned to this arc. The text is then scanned, first to the right and then to the left, according to the conditions in the arcs to the right and to the left of the anchoring arc. An automaton succeeds if all its arcs match some input text. This means that a particular text pattern, or an error, has been detected. We use automata both to extract syntactic information from the text (these are pre-processing automata) and to detect errors in the text (error detection automata). Because the automata are all finite-state automata, and hence equivalent to deterministic automata (Partee *et al.*, 1990), non-determinism is not a problem.

The checker does not attempt to parse sentences fully. Instead, preprocessing automata analyse certain islands which can later be used by error detection automata. This island parsing approach, which is followed by an error search, has at least two advantages over the more traditional methods based on a full parse. First, second

language texts may contain many errors (several within the same sentence) which can cause the parsing process to fail completely. Even if the texts are purged of these errors during a preceding stage, it is not clear whether today's parsing algorithms are as yet suitable in this situation (see Cornu, 1992). Second, automata can be programmed to identify specific situations where errors occur and thus they do not have to take care of non-relevant parts of the text.

4.1 Linguistic information used by the automata

Due to the extremely wide range of errors that the error detection automata have to detect, their formalism was designed so as to let them have access to a full spectrum of linguistic information available in the system, such as canonical and inflected word forms as well as morphological, lexical, syntactic and some semantic features. Some rules match specific word sequences, while others use noun phrases or prepositional phrases. The linguist writing the automaton determines the type of data required to identify each error.

Error detection automata rely on four sources of information:

(a) The main dictionary, which includes the syntactic categories of each word, some morphological information and a few syntactic and semantic features (such as whether a noun is countable, a verb is transitive, a pronoun is possessive, etc.). The dictionary is an extract of CELEX (Burnage, 1990) and contains approximately 4800 canonical forms corresponding to more than 20,000 inflected forms.
(b) A set of lists that contain words which share identical semantic or syntactic features. Some lists are very general such as, for example, the one containing words with the feature PLACE (e.g. 'office', 'Switzerland') or TIME (e.g. 'Wednesday', 'week'). Others are more specific such as the one containing verbs that are not usually used in a continuous tense, like 'to last' and 'to seem'.
(c) Noun phrases (simple and complex), marked with time and location when appropriate, as well as head, number and person features.
(d) Tense, mode and aspect features for all verb forms.

Four software modules obtain the above-mentioned information and prepare it for the detection automata:

1. A tokenizer (or segmenter) divides the input text into words, numbers and punctuation marks. Each word can then be looked up in the dictionary. For research purposes, we correct misspelled words by hand. A spelling correction module would normally be activated at this stage to handle the words not found in the dictionary.
2. A neural-network algorithm eliminates the non-relevant syntactic categories for multi-category words and keeps only one (see Bodmer, 1994). For example, in 'The can was rusted', 'can' is categorized as a noun and not as a verb.
3. A noun phrase parser identifies simple non-recursive noun phrases. It is based on an algorithm of the type presented in Church (1988), and was trained on part

of our error corpus where the NPs had been marked manually. The correct syntactic category is identified with a high degree of success by the module.

4. Pre-processing automata build complex noun phrases (e.g. 'the last day of the month') and determine the noun phrase features described above. They also analyse all verb forms and assign tense, voice and aspect features to the main verbs.

4.2 Automata formalism

A number of constraints were placed on the metalanguage for specifying the pre-processing and error detection automata. The formalism had to be powerful enough to allow access to the wide range of information that was available but, at the same time, it had to be transparent to the linguists developing the rules. In addition, the automata execution module had to be easily integrated into an operating environment such as Microsoft Windows. The result is a formalism very similar to the one used by Prolog where the individual arcs that form the automata are enclosed in square brackets ('[...]'). The conditions on the input text are separated by commas and the features are written in plain text or with mnemonic abbreviations. The automata are not executed in Prolog, but by a module written in C that allows for fast execution and easy integration into the user-interface development tools.

Just as in Winograd's approach, the automata use registers to carry information between arcs. These registers also carry all the information needed by the correction module. Automaton 1 shows the simplified form of a pre-processing automaton that identifies conditional verbs and assigns a series of features to the main verb (as in 'You should be dancing'):

Automaton1
@ [CAT = V, (IF = 'would'|IF = 'should'|IF = 'could'|IF = 'might')]
(AFF = > $S)
[IF = 'not', NEG = > $S]*1
[IF = 'be']
[CAT = V, TNS = ING, VF_TENSE: = COND, VF_ASPECT: = CONT,
VF_VOICE: = ACT, VF_STATUS: = $S]

The following takes place in this automaton:

(a) The first arc, which is also the anchoring arc (@), matches the words 'would', 'should', 'could' or 'might'.
(b) An action then occurs (it is written between parentheses to indicate that it does not match a word in the input text). If the first arc has succeeded, the action in the second line is executed, namely, putting the value AFF (for 'affirmative') into register $S.
(c) The second arc (third line) optionally matches the word 'not'. If that word is present in this position in the text, then the value of register $S is changed to NEG (for 'negative'). At this stage, therefore, the value of $S is NEG if the word 'not' is present in the input text, and AFF if it is not present.
(d) The third arc simply matches the word 'be'.

(e) The fourth arc performs two operations. First, it checks that the word is a verb in the present participle form ('-ing' form). Then it defines the four attributes VF_TENSE, VF_ASPECT, VF_VOICE and VF_STATUS for the word which is identified as the main verb. Note that the value of $S has been carried through the automaton and that it is used to set the value of the feature VF_STATUS.

Arcs that match noun phrases have the format '⟨NP (conditions)⟩'. The conditions specified inside the delimiters apply to the noun phrase itself. This may include matching the entire noun phrase or looking for specific words. The two examples below illustrate how these types of arcs are used in detection automata.

Automaton2
⟨NP⟩
@ [CAT = ADV, +ADV_POS]
[CAT = V, (+MODAL | +AUX)] +2
[CAT = V]

This simplified automaton locates errors in constructions such as '*He never may come back' where the adverb is misplaced.

(a) The anchoring arc (preceded by @) matches adverbs included in the ADV_POS list, such as 'frequently', 'often', 'now'.
(b) A noun phrase is looked for to the left of the anchoring arc (without any special conditions).
(c) One or two modal or auxiliary verbs followed by a verb are then matched on the right-hand side of the anchoring arc.

Automaton3
@ [IF = 'during']
[IF = 'almost' | IF = 'nearly' | IF = 'about'] *1
⟨NP @ [+NP_HEAD, NBR = PLUR, +TEMP]⟩

In this second example, also presented in its simplified form, the automaton identifies the wrong preposition in prepositional phrases such as '*during five hours'.

(a) The first arc matches the word 'during'.
(b) The second arc optionally matches one of three adverbs.
(c) The third arc matches a noun phrase and checks if its head is plural and belongs to the TEMP list which contains temporal words.

4.3 Error detection

So far, we have seen that automata are used for a number of purposes. Their formalism allows them to identify the main verb and assign tense parameters to it (Automaton 1), detect the wrong position of adverbs (Automaton 2) or detect the wrong preposition in prepositional phrases (Automaton 3). To take advantage of this flexibility, we developed a schedule program, which defines the role of each

automaton within the error detection process. Thus, the process of identifying a specific error can be broken down into a sequence of steps, such as extracting information (e.g. the tense of the main verb), testing the syntactic environment (is a particular noun phrase the subject of the verb?) and undertaking the actual error detection (do the main verb's number and person features match the subject's?). The schedule program also allows for local errors to be checked before those that involve larger sequences of words.

The schedule program is divided into two sections: the pre-processing section, where attributes are assigned to words, noun phrases and prepositional phrases, and the detection section, which includes all error detection automata. The pre-processing section includes automata such as Automaton 1, whose task is to find verbs in the conditional. This pre-processing section mainly contains lists of automata grouped according to their function. Here are two examples of such lists (the names of the automata are given in brackets):

List1
{NP_HEAD}

List2
{NP_NBR_SING_QUANT NP_NBR_GEN
NP_NBR_SING_PRON NP_NBR_PLUR_PRON

The first list only contains the automaton NP_HEAD which determines the head of noun phrases, and the second list contains four automata that determine the number of noun phrases.

In the detection section of the schedule program, the rule developer can use two approaches to organize the automata. The first consists of simple lists, like the ones in the pre-processing section. For example:

List3
510 {N_AGR_PLUR_SING_1 N_AGR_PLUR_SING_2
N_AGR_PLUR_SING_3

Here the number at the beginning (510) identifies the list. It links the grammar corrector's options to the automata group. The user can thus selectively activate or deactivate the automata dealing with specific types of errors (subject-verb agreement, use of tenses, prepositions, etc.). The second approach is a scheduling tree, used when the triggering of some detection automata depends upon a given sentence structure. In this case, one or more filter automata determine if a certain structure is present in the text. If it is, the corresponding set of detection automata is triggered. The following example illustrates this mechanism:

List4
FILTER
{DAT_V_FIL_1 DAT_V_FIL_2
($R = R1) {SUFLU_PREP_DATV_1
($R = R2) {PREP_DATV STRUCT_DAT_V1 STRUCT_DAT_V2
END

Here the automata list containing DAT_V_FIL_1 and DAT_V_FIL_2 is executed first. The two filter automata trigger on dative verbs that have certain characteristics. If one of these two automata succeeds, it sets the value of register $R to R1, to R2 or to some other value. This determines which of the two other lists should be executed next. Both of these lists deal with the presence or absence of a preposition in the verb's complement. If the value of register $R is R1 then the list with SUFLU_PREP_DATV_1 is executed. If the value is R2 then the second list is executed. If the value is neither R1 nor R2 then processing continues normally.

4.4 Error correction

Once an error detection automaton has succeeded, indicating the presence of an error in the text, the user is informed. This is done by displaying a message in the correction window and presenting the user with a set of choices. We have developed a set of nine standardized user-interface dialog procedures. Each one corresponds to a specific error correction situation. For example, Dialog Procedure 2 is applied when the correction involves replacing a word or a set of words with one or more words, and Dialog Procedure 3 involves inserting a new word.

The dialog procedures all require a certain number of parameters which are attached to the error detection automata. They include:

(a) A dialog procedure number which specifies which dialog procedure is to be applied.
(b) A message which contains the text to be presented to the user.
(c) A word list which contains all the words proposed to the user. This parameter is used by the replacement and insertion dialog procedures.
(d) A linguistic help which identifies the help screen to be displayed when the user presses the Help button.

In addition to these parameters, pre-defined registers are used to indicate where the insertion, replacement, deletion or permutation must occur within the text. The value of these registers is also used to insert parts of the input text into the error message presented to the user.

Automaton 4 shows how a missing 's' in NPs such as '*these student' is identified.

```
Automaton4
⟨NP @ [IF = 'these', CURPOS =⟩$L, CURPOS =⟩$R]
[CAT = N, NP_POS = NP_LAS, ~ NUMB, ~ PLUR_N, NBR = SING,
  CURPOS =⟩$Z, IF =⟩$B] ⟩
{
DIALOG PROCEDURE 2
MESSAGE 'Il semble qu'il y ait un problème d'accord dans la séquence \u\($L-
  $Z)\v. Le mot \uthese\v doit être suivi d'un nom au pluriel. Nous vous suggérons
  de mettre \u\$B\v au pluriel ou alors de remplacer \uthese\v par \uthis\v.'
LIST 'this'
}
```

As can be seen, the French-speaking user receives the message in French. Here is an English translation of this particular message: 'There seems to be an agreement error in the sequence \u\($L-$Z)\v. The word \uthese\v must be followed by a plural noun. We suggest that you use the plural form of \u\$B\v or replace \uthese\v with \uthis\v.'

Here is what happens in this automaton:

(a) Registers $L and $R (for left and right) identify where the replacement should occur if the user selects the Replace button. Register $Z identifies the right edge of the noun phrase where the error occurred and register $B contains the noun in question.
(b) Registers $L, $Z and $B are used in the message that appears on the screen for the user. '($L-$Z)' is replaced by the segment of the input text between positions $L and $Z (i.e. '*these student'), and '$B' is replaced by 'student'.
(c) When presented to the user, the parts of the message between the control characters \u and \v are underlined and appear in red.
(d) DIALOG PROCEDURE 2 indicates that this particular procedure is to be applied.
(e) LIST 'this' indicates that a list with the single word 'this' will be presented to the user. When the user presses 'Replace', the word selected in the list replaces the word in position $L (i.e. 'this' replaces 'these'). In other situations, the list may contain more than one word.

4.5 User interaction

Three special dialog procedures (7, 8 and 9) are used to dialog with the user. This is sometimes required when it is not possible to determine all sentence attributes automatically. Normally, when no user interaction is possible, the rule designer is faced with a dilemma – either not process these types of errors, or try to tackle them, but with the risk of overflagging, that is detecting an error when there is none. With these dialog procedures, however, the rule designer can have the system ask the user for some information and then proceed based on the answer that has been given.

Consider Automaton 5 (again simplified), which identifies erroneous sequences such as '*the more great', i.e. 'the'+'more'+an adjective belonging to the list of adjectives forming their comparatives by adding -er (+ADJ_ER).

Automaton5
[IF ≠ 'all']
[IF = 'the', CURPOS = >$L]
@ [IF = 'more']
[CAT = A, +ADJ_ER, CURPOS = >$R]
{
DIALOG PROCEDURE 7
MESSAGE 'La séquence \u\($L-$R)\v ne semble pas correcte. Vouliez-vous exprimer (A) un comparatif, ex. ''plus beau que'' ou (B) un superlatif, ex. ''le plus beau''?'
}

Translation of the message: 'The sequence \u\($L-$R)\v looks incorrect. Did you want to express (A) the comparative, as in "more beautiful than" or (B) the superlative, as in "the most beautiful"?'

When this type of sequence has been identified, the type of action required depends upon whether the user wanted to use a comparative or a superlative adjective. The system needs to tell the user to replace '*more great' with 'greater' in the first case and with '(the) greatest' in the second. The scheduling program is as follows:

```
List5
FILTER FIND_THE_MORE_ADJ
( ) THE_MORE_ADJ
($Q = R1) THE_MORE_ADJ_COMP
($Q = R2) THE_MORE_ADJ_SUP
END
```

The automaton FIND_THE_MORE_ADJ first identifies the syntactic environment. The THE_MORE_ADJ automaton is then executed and the message it contains is displayed. The user is presented with buttons labelled A and B. The user's input sets the value of register $Q to R1 or R2 depending on which button s/he presses. Control then returns to the schedule program which executes either automaton THE_MORE_ADJ_COMP (if A was pressed) or automaton THE_MORE_ADJ_SUP (if B was pressed).

5 Evaluation of the grammar checker

Although our grammar checker is still at the prototype stage, we decided to undertake a first evaluation of its performance. We used an evaluation kit developed by Tschichold (1991) and compared our prototype to three commercial grammar checkers: Correct Grammar for Macintosh (version 2.0 of the monolingual English grammar checker developed by Lifetree Software and Houghton Mifflin), Grammatik for Macintosh (the English for French users 4.2 version of the Reference Software grammar checker) and WinProof for PC (version 4.0 of Lexpertise Linguistic Software's English grammar checker for French speakers, which is based on Houghton Mifflin's CorrecText to which has been added specific second language error detection capabilities). It should be noted that we did not include our writing aids or our problem word highlighter in the evaluation nor did we examine other aspects such as user interface, dictionary access, compatibility with word processors and so on. The evaluation kit we used contains a number of tests that examine the kind of errors made by French native speakers when writing in English. These have been classified into eleven categories: verb form (*She regret it), verb tense and aspect (e.g. *She is born in Paris in 1956), nouns (e.g. *I work hard all the times), determiners (e.g. *He waited all a week), adjectives and adverbs (e.g. *Let's help these poors animals), prepositions (e.g. *She resembles to her father), pronouns (*They found pleasant to see you), conjunctions and inversion (e.g. *He said that no), lexical errors (*This remembers me of my brother), style (e.g. She says she needs diapers and a

pram) and continuous text (which contains errors taken from all categories). Additional examples of test sentences are given in Appendix 1.

All categories contain 10 sentences with one error per sentence except for the last category (continuous text) which contains 55 sentences. There are three possible outcomes for each error, as can be seen in Table 1:

(a) it is detected as an error (we have called this a detection; cell A in the table),
(b) it is detected as a potential error (which we call a warning; cell B),
(c) it is not detected (no detection; cell C).

Table 1. *Possible outcomes when an error and correct prose* (*No error*) *are processed by a grammar checker. The score given to each outcome by the evaluation kit are in parentheses*

Error status	Detection	Warning	No detection
Error	A (+2)	B (+1)	C (−1)
No error	D (−2)	E (−1)	F (0)

As for the correct material in the rest of the sentence, there are also three possibilities:

(d) an 'error' is detected where there is none (cell D),
(e) a warning message is given when the text is correct (cell E),
(f) no message is given (i.e. there is no error and no detection; cell F).

Thus, this test takes into account errors that are detected and not detected as well as overflaggings, that is, false error detections and warnings (type D and E above). This is particularly important as overflaggings can create much more confusion for non-native speakers than for native speakers of a language.

A score, positive or negative, was given to each of the six outcomes, as can be seen in Table 1. An appropriate detection or warning gets a positive score, whereas an error that is not detected or detected erroneously (overflagging) gets a negative score. We ran the test on all four grammar checkers, the three commercial checkers and our own, and for each category, we subtracted the score obtained by each of the commercial checkers from our prototype's score. This showed how we fared. The results are presented in Table 2. A positive number indicates how much better our prototype does and a negative number how much worse. A yardstick to help understand these differences is to keep in mind that 2 points in favor of our prototype (a positive number in the table therefore) means either that we detected an error which another grammar checker did not detect or that we did not detect a non-error that the other checker did detect. On the contrary, a difference of minus 2 points means either that we did not detect an error detected by another checker or that we overflagged a non-error which it did not. As can be seen in the table, our prototype does better on 30 out of the 33 possibilities (20 of those by 3 points or more), does as well twice and does worse only once. Although the differences are never above 10

(with the exception of the categories Style and Continuous text), this is an encouraging first result as our checker is only a prototype that needs to be completed.

Table 2. *Difference, in scores, between three commercial grammar checkers and the prototype for each of* 11 *evaluation categories. A positive number indicates how much better the prototype fares and a negative number how much worse*

Categories	Correct Grammar	Grammatik	Winproof
Verb forms	7	9	0
Verb tense and aspect	6	7	6
Nouns	−3	0	2
Determiners	4	3	7
Adjectives and adverbs	6	2	4
Prepositions	5	1	4
Pronouns	2	2	3
Conjuctions and inversion	2	2	6
Lexical errors	1	1	1
Style	4	15	34
Continuous text	9	9	28
Total	**43**	**51**	**95**

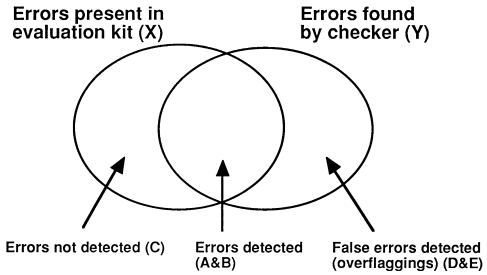

Fig. 1. Visual representation of the error and false error categories used to compute the error detection and error overflagging scores of each checker.

Another way of comparing our prototype to existing checkers is to examine error detection and error overflagging scores separately. This will also help us understand the results presented already. As can be seen in Figure 1, we considered for each checker the total number of errors present in the 11 categories of the evaluation kit (represented by X in the figure) and all the errors that it found (represented by Y). Next we calculated the errors not detected (C in Figure 1 and Table 1), the errors detected (A & B; we gave the same status to detections and warnings) and the false errors detected (that is, the overflaggings; D & E; again we gave the same status to detections and warnings). We then computed, for each checker, a detection score by dividing the errors detected (A & B) by the errors present in the evaluation kit (X),

and an overflagging score by dividing the false errors detected (D & E) by the errors found by the checker (Y). Both measures were then expressed as percentages.

As can be seen in Table 3, the error detection scores are rather low for all of the grammar checkers evaluated. WinProof does the best and our prototype is in second position followed quite closely by Grammatik and Correct Grammar. Given that our prototype has a reduced lexicon and a limited number of automata, this result is quite encouraging. What is particularly rewarding is the error overflagging score for the prototype which is much lower than that of the commercial grammar checkers. This is in large part due to our policy of not flagging potential errors in the grammar checker but letting the user examine them with the problem word highlighter. This last result explains why we usually do better than the other checkers in the overall scores presented in Table 2 which take into account both error detection and non-error overflagging.

In sum, at this early stage of its development, our prototype does quite well against well established commercial products.

Table 3. *Error detection and error overflagging scores for the prototype and three commercial grammar checkers*

Measure	Prototype (%)	Correct grammar (%)	Grammatik (%)	WinProof (%)
Error detection	14·5	10·5	12·5	34
Error overflagging	43·5	74	75	76

6 Conclusion

In this paper, we have presented the prototype of a second language writing tool for French speakers writing in English. It is made up of a set of writing aids, a problem word highlighter and a grammar checker, and it is characterized by a number of modules and principles that ensure that non-native speakers receive the help they need when writing in their second language. Among the modules, we should note the presence of various aids that can be used during the writing process and a grammar checker that is tuned to the kind of errors made by such writers. Among the principles, we should stress limiting overflaggings (these can have serious consequences for non-native speakers) and employing user interaction to help the error detection and correction process.

So far, some three years of work by about eight researchers (working at various percentages of time) has been put into the prototype. In addition to the various processing modules already built and described in this paper, the prototype contains a large number of pre-processing and detection automata, a number of relatively well developed writing aids (on-line grammar, list of difficult words, verb conjugator) and a finished version of the problem word highlighter. Future work will involve increasing the size of the grammar checker dictionary, adding automata to cover more errors, finishing some of the writing aids, and adding a monolingual spell

checker as well as commercial dictionaries (monolingual English and bilingual French-English). We should note that from the start, our system was designed for use outside the laboratory. Thus, the error processing technology that we have implemented allows for all components to be expanded without risking a degradation in quality. Once completed, the prototype will be ready for commercial development which will include, among other things, integrating it into existing word processors and evaluating it thoroughly with second language users.

Appendix 1

Additional test sentences used in the evaluation kit.

1. Verb form
By which road did you came?
He was died last year.

2. Verb tense and aspect
Every day I am learning things I never knew before.
I lived there for ten years when I left for Geneva.

3. Nouns
It was one of those hot day in June.
The unions are trying to get industry to accept a thirty-five-hours week.

4. Determiners
You are not expected to make noise here.
At the ceremony about hundred people were present.

5. Adjectives and adverbs
This box is more heavy than that one.
This book is twice bigger than the one I bought.

6. Prepositions
My teacher explained it me four times.
Her sister is married with an army officer.

7. Pronouns
The vacuum cleaner makes easy to clean the house.
Can I wash the hands before we start?

8. Conjunctions and inversion
Eight thousand thirty-seven
That what some politicians say to get votes is amazing.

9. Lexical errors
This remembers me of my brother.
We got into the train and I put my bag upstairs on the rack.

10. Style
How do you spell labour and color?
She says she needs diapers and a pram.

References

Allen, J. (1987) *Natural Language Understanding*. Menlo Park, CA: Benjamin/Cummings.

Barchan, J., Woodmansee, B. and Yazdani, M. (1986) A PROLOG-based tool for French grammar analysis. *Instructional Science* **14**:21–48.

Bodmer, F. (1994) DELENE: un désambiguïsateur lexical neuronal pour textes en langue seconde. *TRANEL* (*Travaux neuchâtelois de linguistique*) **21**: 247–263.

Brehony, T. and Ryan, K. (1994) Francophone stylistic grammar checking (FSGC) using link grammars. *Computer Assisted Language Learning* **7**(3): 257–269.

Burnage, G. (1990) *CELEX – A Guide for Users*. Centre for Lexical Information, University of Nijmegen, The Netherlands.

Catt, M. (1988) *Intelligent diagnosis of ungrammaticality in computer-assisted language instruction*. University of Toronto, Technical Report CSRI-218.

Church, K. (1988) A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of the Second Conference on Applied Natural Language Processing*. Austin, Texas.

Corder, S. (1967) The significance of learners' errors. *International Review of Applied Linguistics* **5**: 161–170.

Cornu, E. (1992) *The importance of Linguistic Theories in Grammar Checking*. Workshop on natural language processing: first and second language correction of written texts, SGAICO, Neuchâtel, Switzerland.

Granger, S. and Meunier, F. (1994) Towards a grammar checker for learners of English. In U. Fries, G. Tottie and P. Schneider (eds.) *Creating and Using English Language Corpora*. Amsterdam: Rodopi.

Grosjean, F. (1982) *Life with Two Languages*: *An Introduction to Bilingualism*. Cambridge, MA: Harvard University Press.

Guillemin-Flescher, J. (1981) *Syntaxe comparée du français et de l'anglais*. Paris: Ophrys.

Kübler, N. (1992) Verbes de transfert en français et en anglais. *Linguisticae Investigationes* **16**(1): 61–97.

Kübler, N. (1995) *L'automatisation de la correction d'erreurs syntaxiques*: *Application aux verbes de transfert en anglais pour francophones*. PhD thesis, Publications de l'Institut Gaspard Monge, Vol 6, Université de Marne La Vallée, France.

Partee, B., ter Meulen, A. and Wall, R. (1990) *Mathematical Methods in Linguistics*. Dordrecht: Kluwer.

Payette, J. and Hirst, G. (1992) An intelligent computer-assistant for stylistic instruction. *Computers and the Humanities* **26**: 87–120.

Selinker, L. (1972) Interlanguage. *International Review of Applied Linguistics* **10**(3): 209–231.

Selinker, L. (1992) *Rediscovering Interlanguage*. London: Longman.

Silberztein, M. (1993) *Dictionnaires électroniques et analyse automatique de textes*. Paris: Masson.

Thurmair, G. (1990) Parsing for grammar and style checking. *COLING*: 365–370.

Tschichold, C. (1991) *The Evaluation of Computer-Assisted Writing Tools for Non-native Speakers of English*. Englisches Seminar, Universität Basel.

Vinay, J. and Darbelnet, J. (1977) *Stylistique comparée du français et de l'anglais*. Paris: Didier.

Winograd, T. (1983) *Language as a Cognitive Process*: *Syntax*. Reading, MA: Addison-Wesley.

Yazdani, M. (1993) *Multilingual Multimedia*: *Bridging the Language Barrier with Intelligent Systems*. Oxford: Intellect.