TRANEL (Travaux neuchâtelois de linguistique), 1994, 21, 181-194.

Automatic correction of French prose written by English native speakers: An LFG approach

Etienne Cornu

Abstract

We present a parser based on Lexical Functional Grammar (LFG) that is capable of detecting and correcting errors produced by English native speakers when writing in French. The errors are syntactic and morpho-syntactic in nature such as subject-verb and past participle agreement, adjective-noun word order, use of subjunctive in subordinate clauses, etc. The system processes a text sequentially, sentence by sentence, and the error-tolerant LFG parser can process any number of errors in the same sentence. A first evaluation shows that the system can correct some 70% of the errors in a small corpus.

1. Introduction

During the last decade, much effort has been put into developing computer programs that can correct errors in written text. Examples of such programs are the spell checkers that equip virtually every word processing software. If these programs seem to satisfy a very large number of users writing in their native language, this is not the case for those who need to write in a second language. For example, non-native writers of French or German make the common mistake of using the wrong gender with articles or adjectives. To help them detect, and possibly correct, this type of error, a program operating with a different type of knowledge is needed.

2. Background

A large variety of error detection and correction tools have been developed for English as a first language. They range from spell checkers that generate word alternatives using phonological rules to style checkers based on stylistic theory (Payette, 1990). One of the most advanced examples of such tools is the commercial product Correct Grammar. It performs a complete parse of each sentence using mechanisms that analyze even complex text structures, abbreviations, quotation marks, etc.

However, even if Correct Grammar represents quite an achievement, its usefulness is still questionable (Dobrin, 1990). Some grammar rules implemented in Correct Grammar are not normally observed by users, simply because they have different writing styles, whereas other rules are irrelevant because they are never broken.

The situation in the area of second language correction is not as developed as in first language checking. Although linguistic research in contrastive analysis and more recently in interlanguage theory (Corder, 1967; Selinker, 1972; Richards, 1972; Flynn, 1988) has produced many good descriptions and classifications of second language errors, little has been done to implement tools that will detect and correct these errors.

3. System Description

The aim of our research is to study the mechanisms involved in dealing with errors occurring specifically at the levels of syntax and morpho-syntax. To achieve this, we have implemented a parser based on Lexical Functional Grammar (Bresnan, 1982), later referred to as LFG. This parser is capable of detecting and correcting errors in written French produced by English native speakers. The errors that are processed fall into the following domains: subject-verb agreement, adjective-noun word order, noun phrase agreement, past participle rule, use of the subjunctive in subordinate clauses, subcategorization of selected verbs and usage of *à*, *au*, *aux*, *du* and *des*.

The system processes a text sequentially, sentence by sentence, and displays warning messages and suggestions for corrections when it detects an error. The system's error-tolerant LFG parser allows it to process any number of errors in the same sentence or sentence segment. It should be noted that we have put little emphasis on the interaction with the user, as the purpose of our research was to investigate the feasibility of using LFG as a basis for automatic error processing rather than to develop a ready-to-use second language writing tool.

4. Error detection using LFG

In LFG, the grammatical relations between words are encoded in functional structures ("f-structures"), which consist of ordered attribute-

value pairs. LFG imposes three grammaticality conditions on the contents of functional structures: uniqueness, coherence and completeness. The uniqueness condition specifies that an attribute in an f-structure may have at most one value. For example, the attribute 'number' may not be 'singular' and 'plural' at the same time. The coherence and completeness conditions apply to the subcategorization arguments of verbs and nouns. Together they impose a one-to-one correspondence between the functions present in an f-structure and the subcategorization list of the head of the f-structure.

The error detection mechanism of our system relies solely on these grammaticality conditions. The following examples show how each one is used.

1- Uniqueness

LFG theory uses common attributes such as 'gender', 'number', 'person' and 'case' to impose certain constraints on the text. One way to add constraints, and thus to detect additional errors, is to insert additional attributes. For example, this can be done to check the use of prepositions *de* and *à* in sentences such as *Il me permet de partir* (He allows me to leave) and *Il m'autorise à partir* (He authorizes me to leave). The following conditions are attached to the lexical entries of the above examples:

permet	(↑comp-prep) = prep-de
autorise	(↑comp-prep) = prep-à
de	(↑comp-prep) = prep-de
à	(↑comp-prep) = prep-à

These force the use of the preposition *de* with *permet* and *à* with *autorise* in the same way that the attributes 'number' and 'person' ensure the proper subject-verb agreement. However, this approach is insufficient in more complicated cases: the past participle rule in French is a typical example where conditions apply at more than one level. The gender and number of the past participle depend, amongst other things, on which auxiliary verb is used (*être* or *avoir*), on the position of the direct object

relative to the verb and on the gender and number of the subject or the direct object. In this case, the solution consists in adding not just one attribute to a lexical entry but a series of if-then rules that insert the correct constraints in the f-structure. Consider the sentence *Il les a achetés hier* (He bought them yesterday). The lexical entries for *achetés* and *a* are as follows:

achetés	(↑number) = plural	
	(↑gender) = masculine	
a	if	(↑obj position) is after-verb
	then	(↑number) = singular
		(↑gender) = masculine
	else	(↑number) = (↑obj number)
		(↑gender) = (↑obj gender)

(If the direct object is placed after the verb, the past participle is masculine singular, but if it is placed before the verb, as is the case in our example, the past participle takes the number and gender of the direct object.)

The value for the attribute 'position' is inserted in the f-structure by the phrase structure rule that was instantiated at the verb phrase level. The following simplified example shows how this is done:

VP	:==>	V	NP	PP
			(↑obj)= ↓	(↑obj2)= ↓
			(↑obj position)	
			= after-verb	

This mechanism appears in all situations where the order of sentence components is important, as when you have to indicate whether an adjective precedes or follows the noun in a noun phrase.

2- Coherence and completeness

Some verbs, such as *téléphoner* (to phone) and *donner* (to give), cause a lot of confusion among English native speakers writing in French as their complements are constructed differently in the two languages. In English, you phone someone whereas in French you phone *to* someone

(*téléphoner à quelqu'un*). This is obviously a case where, if the wrong construction is used, both the coherence and the completeness conditions are violated. Because of the large number of different phrasal constructions that arise in everyday texts, it would be unreasonable to flag an error every time one of these two conditions failed. Thus, instead of storing a comprehensive set of complement combinations, the system only flags an error when a specific situation occurs, such as:

IF	the coherence or/and the completeness condition is not satisfied
AND	the verb in question is a member of a specific list of misleading verbs
AND	a specific combination of complements occurs
THEN	an error is detected.

For the verb *téléphoner*, for example, an error is indicated only if a direct object is found in the f-structure instead of an indirect object. If the coherence or the completeness conditions fail for a verb not in the list, no action is taken.

5. Organization of the program

The central elements used for error detection and correction are the f-structures representing the input text. These are generated by an error-tolerant parser that produces one f-structure for each interpretation of the text and, at the end, discards all f-structures but one. The error correction stage then checks this f-structure for inconsistencies, determines the cause of the errors it may contain and corrects these when possible.

5.1. Error-tolerant parsing

Standard LFG parsers usually build f-structures in parallel with phrase structure trees called constituent structures ("c-structures"), so as to control the explosion of the number of solutions (Block and Hunze, 1986). With such methods, partial f-structures are discarded during parsing as soon as one of the grammaticality conditions fails. A different approach is necessary when the text contains errors (as in our case) as the grammaticality conditions fail for every single solution and no f-structure can be produced. Another difficulty encountered at this stage is that only

a few sentences can be fully parsed using today's methods. The presence of errors only amplifies this situation. In order to contain these problems, we have divided the parsing stage into two steps:

Step 1 - Construction of c-structures and segmentation

A chart parser is used to apply phrase structure rules to the text. If no complete c-structure can be created, a segmentation algorithm extracts islands from the chart (usually noun phrases and verb phrases) which can be processed separately.

Step 2 - Unification

Instead of discarding the f-structures that contain inconsistencies, the unification algorithm maintains a list of each attribute's values and their source, which can be a lexical item or a c-structure node equation. An error is recorded as soon as a list contains two different values, i.e. when the uniqueness condition is not satisfied. The coherence and completeness conditions are only checked after the f-structures have been built completely.

The final operation executed at this stage consists in counting the errors in each f-structure. Only the one with the fewest errors is kept and corrected.

5.2. Error processing

Given the f-structure produced by the parser, the goal of the error correction stage is to put the input text into grammatical form. When this is not possible, the error corrector produces a warning message without proposing a correct form. Regardless of the nature of the error in the text, each error is represented in the f-structure either by an attribute with ambiguous values or by a PRED attribute for which the subcategorization arguments do not match. Thus, instead of processing the text word by word, the error processing mechanism executes one complete correction procedure for each flagged attribute. Because every attribute is subject to having ambiguous values, one procedure is attached to every single attribute used by the grammar.

The behavior of the correction procedures varies according to the type of errors covered.

5.2.1. Ambiguous attributes related to morphological elements

Morphological errors occur, for example, when the gender of an article or an adjective is wrong, or when the verb is at the wrong person or at the wrong mode. Most error correction systems in this case only give a warning message to indicate the noun phrase or sentence fragment in which the error occurred. We have gone one step further in that the wrong words are actually corrected and replaced in the text. But this approach requires more information than what is usually encoded in the lexicon.

Consider the following two examples:

(a) * *Le maison* (the house) (b) * *Les maison* (the houses)

In (a) the gender of the article should be feminine instead of masculine, and in (b) the number of the noun should be plural. In both cases, the data in the f-structure only indicate a mismatch between the values of one attribute. To fill this gap, the error corrector uses a set of lexical rules to decide which value to select. Here are extracts of such rules:

- The gender of nouns takes precedence.
- The number of articles takes precedence.
- The past participle number and gender introduced by the auxiliary verb take precedence.

Given this information, the error correction procedures for morphological attributes execute the following operations:

(1) Find the reference value of the attribute.

(2) Find all the words that have a different value for this attribute.

(3) Search the lexicon for the new word form.

(4) Edit the text or show the new word.

In example (a) above (**Le maison*), the ambiguous attribute is 'gender'. Step (1) finds that the reference value is that of the noun, *maison*. Step (2) identifies that the article has a different value. Step (3) looks for the feminine equivalent of *le* in the lexicon and finds *la*. Finally, step (4) replaces *le* with *la*.

The same mechanism is also used to verify the past participle rule. The if-then rule shown for the auxiliary *avoir* in section 4 above introduces reference values for gender and number, and if they don't match the values of the past participle, the lexicon is accessed in order to find the new word.

5.2.2. Ambiguous attributes not related to morphological elements

The case of ambiguous attributes not related to morphological errors is similar to the previous one in that a reference value for the attribute is present, as in **Il me permet à partir* (He allows me to leave). Here, the attribute 'comp-prep' received the wrong value 'prep-a' from the word *à* instead of 'prep-de', which is the reference value obtained from *permettre*. In such cases, the correction procedure simply displays a warning message indicating that the main verb requires the preposition *de*.

5.2.3. Subcategorization errors

As mentioned earlier, the procedures attached to subcategorization errors only check for specific conditions to be present in the f-structure. The following example shows how the subcategorization for the verb *téléphoner* is verified.

IF pred = *téléphoner* THEN

IF obj is present

AND obj2 is not present

THEN print a warning message of the type
"use *téléphoner à quelqu'un* "

In all other cases, no action is taken.

This basic correction method can be improved somewhat when the direct object is a pronoun, as in the sentence * *Il la téléphone* (He calls her). A check for a pronoun can be added in the list of conditions, and the method used for morphological attributes (finding the reference value) can be applied to replace the accusative form of the pronoun with the dative (*lui*).

6. Evaluation

The aim behind the evaluation we conducted was to determine the extent to which LFG-based algorithms could be used in second language error correction systems. The corpus we used consisted of 165 sentences obtained mainly from grammar books for learners of French and illustrating the problems related to each of our 11 error categories. Two thirds of the sentences contained an error, the others were grammatically correct.

The error categories were the following:

1. Subject-verb agreement
2. Noun phrase agreement
3. Confusion between infinitive and past participle
4. Past participle rule with *être*
5. Past participle rule with *avoir*
6. Use of subjunctive in subordinate clauses
7. Subcategorization of verbs
8. Adjective-noun word order
9. Use of prepositions *de* and *à* with infinitive verbs
10. Confusion between *à* and *a*
11. Use of *à*, *au*, *aux*, *du* and *des* (preposition+article)

Our corpus was preferred to a set of 'real' texts (such as essays, translations or reports written in French by native speakers of English) because we did not have a lexicon or a set of grammar rules to cover such

texts. Also, the removal of the many errors that were not dealt with by our system was not considered to be appropriate. In addition, some of our error categories may have occurred infrequently.

In order to ensure an objective evaluation, the LFG lexicon was built out of context: the test sentences were segmented into words which were then sorted into alphabetical order before being inserted in the lexicon. The sentences themselves remained hidden until they were submitted to the system.

The results of the evaluation for each category of errors were the following:

category	hit	incorrect detection	miss	false alarm
1. S-V agreement	7	-	3	1
2. NP agreement	8	1	1	5
3. PP vs. inf.	10	-	-	8
4. PP *être*	6	2	2	-
5. PP *avoir*	8	-	2	-
6. subjunctive	4	-	6	-
7. subcateg.	2	1	7	2
8. adj. position	10	-	-	2
9. prep + inf.	6	-	4	1
10. *a* vs. *à*	8	1	1	1
11. prep + art.	8	-	2	-
total	77	5	28	20
	(70%)	(4.5%)	(25.5%)	

Table I - Results for each error category

Out of the 77 hits, 18 (23%) were warnings and 59 (77%) were corrections (the system gave the proper form of the word to replace or to add in the text). As we can see from the results, some error categories are characterized by a high hit rate and a low number of false alarms. Among these, the success of categories 8 (adj. position) and 11 (prep+art) can be largely attributed to the simplicity of the test sentences. A simple pattern matching scheme checking for word pairs would have detected most, if not all, of these errors. Two other categories, 5 (PP-*avoir*) and 10 (*a* vs. *à*), are good examples of cases where the advanced grammatical formalism was used to its full extent.

Two error categories, numbers 2 (NP agreement) and 3 (PP vs. infinitive), are characterized by a high hit rate but also generated a large number of false alarms (13 false alarms altogether). They show what can happen when the error detection mechanism is too loose. Adding detection constraints would reduce the number of false alarms but would also result in fewer detections, which would in turn considerably lower the overall performance of the system.

Finally, some error categories are characterized by a low hit rate. Categories 6 (subjunctive) and 7 (subcateg) are examples where our system did not possess the information required to handle the very specific situations in which these errors occurred.

Another way to look at the results is to use the glass box approach (Palmer and Finin, 1990) where the goal is to determine the behavior of the individual components of the system. In order to achieve this, we have divided the error detection and correction process into four steps and counted how many of the 53 system failures (incorrect detections, misses and false alarms) could be attributed to each one.

We determined that 50 of the 53 failures (94%) were caused by insufficient or inadequate information either in the lexicon (26 cases, 49%) or in the syntactical rules that are used to build the c-structures (24 cases, 45%). Consider the following examples:

a. * *Avoir connus des étrangers est important.*
(To have known strangers is important)

b. * *Je suis content qu'elle vient.*
(I am happy that she is coming)

In (a), the system did not recognize the past infinitive (*avoir connu*). Not only did it fail to detect the error in *connus*, but it also generated two false alarms when it suggested to replace *est* with *sont* and to add an 's' to *important*. In (b), the lexicon did not contain any rule specifying that the expression *être content* required the subjunctive. In the other 3 failures, 2 were due to problems in the selection of the 'best' f-structure (the one that has the least number of errors) and only 1 was caused by an inappropriate correction rule. In this latter case, **cette homme* was replaced with **ce homme*.

These observations show that the algorithms used in the system need complete linguistic data in order to perform well. If we added the missing information, we could expect the performance of the system to improve significantly on the given set of test sentences. There are several problems with this approach however:

1- Modifying the lexicon may produce other false alarms or misses.

2- A different set of test sentences could unveil other weaknesses in the lexicon or the grammar rules.

3- The system's memory and execution time are reaching their limits. Adding more data would require that we adapt some algorithms, thus modifying the behavior of the system (not necessarily for the better).

7. Conclusions

An advantage of using a parser based on LFG is that most of the grammatical rules, such as subject-verb agreement, do not have to be specified explicitly in the program's database. Verifying these rules is done indirectly by applying the well-formedness conditions to f-structures. The use of a modern grammatical formalism also allows the detection of complicated cases, such as the past participle rule and the subjunctive in relative clauses. On the other hand, the data explosion that the system still experiences prevents it from processing long and complicated sentences. We assume that this phenomenon could be

controlled to some extent by comparing and discarding partial solutions during the earlier parsing stage. The evaluation of the system has shown that the algorithms work quite well but that the major bottleneck lies in the amount of lexical and syntactical information needed to create the structures on which the detection and correction algorithms are based. This is a major drawback if we want to use an LFG-type grammar in a commercial application.

We have only demonstrated how sentences containing no more than a single error of a predefined type are handled. But if we consider the large number of spelling, lexical and semantic errors usually contained in texts, we can only assume that the system presented here would be just one element of a general error correction program. How the necessary interaction between our LFG-based algorithms and the other components would occur has yet to be determined.

8. References

ATWELL, E. & S. ELLIOT (1987): "Dealing with ill-formed English text", in: GARSIDE, R. & al. (Eds.), *The Computational Analysis of English*, London, Longman.

BLOCK, H. & R. HUNZE (1986): "Incremental construction of C- and F-structures in a LFG-parser", *Proceedings of COLING*.

BRESNAN, J. (Ed.) (1982): *The Mental Representation of Grammatical Relations*, Cambridge, MA, The MIT Press.

CATT, M. (1988): *Intelligent Diagnosis of Ungrammaticality in Computer-Assisted Language Instruction*, University of Toronto, Technical Report CSRI-218.

CORDER, P. (1967): "The significance of learners' errors", *IRAL*, 5, 161-170.

DOBRIN, D. (1990): "A new grammar checker", *Computers and the Humanities* , 24 (1), 67-80.

FLYNN, S. (Ed.) (1988): *Linguistics in Second Language Acquisition*, Dordrecht, Kluwer Academic Publishers.

PAYETTE, J. (1990): *Intelligent Computer-Assisted Instruction in Syntactic Style*, University of Toronto, Technical report CSRI-247.

PALMER, M. & T. FININ (1990): "Workshop on the evaluation of natural language processing systems", *Computational Linguistics*, 16 (3), 175-81.

RICHARDS, J. (1972): "Social factors, interlanguage and language learning", *Language Learning*, 22, 159-188.

SELINKER, L. (1972): "Interlanguage", *IRAL*, 10 (3), 209-31.

TREMBLAY, J.-P. (1972): *Grammaire comparative du français et de l'anglais*, Québec, Les presses de l'université Laval.