Syntactic error diagnosis in the context of computer assisted language learning

Vandeventer Faltin, Anne

This publication URL: https://archive-ouverte.unige.ch/unige:562
Publication DOI: 10.13097/archive-ouverte/unige:562

UNIVERSITÉ DE GENÈVE FACULTÉ DES LETTRES

Syntactic Error Diagnosis in the context of Computer Assisted Language Learning

THÈSE

présentée à la Faculté des lettres
de l'Université de Genève
pour obtenir le grade de docteur ès lettres
par

Anne Vandeventer Faltin

2003

Contents

Acknowledgments **ix**

1 Introduction **1**
- 1.1 The problem . . . 2
  - 1.1.1 Treatment of ill-formed input . . . 3
  - 1.1.2 The CALL context . . . 4
- 1.2 Objectives . . . 8
  - 1.2.1 Levels of checking . . . 8
  - 1.2.2 Theoretical objectives . . . 10
  - 1.2.3 Practical objectives . . . 10
- 1.3 Methodological approach . . . 12
  - 1.3.1 Selection of errors . . . 12
  - 1.3.2 Reuse and adaptation of a parser . . . 14
  - 1.3.3 Combining diagnosis techniques . . . 16
- 1.4 The FreeText project . . . 18
- 1.5 Structure . . . 19

2 State of the Art **21**
- 2.1 A short history of NLP in CALL . . . 21
  - 2.1.1 The early 1980's . . . 22
  - 2.1.2 The late 1980's . . . 24
  - 2.1.3 The early 1990's . . . 25
  - 2.1.4 The late 1990's and beyond . . . 26
- 2.2 Spell checkers . . . 27
- 2.3 Pattern matching . . . 29
  - 2.3.1 Description . . . 29
  - 2.3.2 Systems using pattern matching . . . 31
- 2.4 Finite State Automata . . . 36
  - 2.4.1 Description . . . 36
  - 2.4.2 The ARCTA prototype . . . 36
- 2.5 Error grammars . . . 38
  - 2.5.1 Description . . . 38

2.5.2 Systems using error grammars 39
2.6 Constraint relaxation 43
2.6.1 Description 44
2.6.2 Systems using constraint relaxation 45
2.7 Other techniques seen in literature 55
2.7.1 Delayed checking 55
2.7.2 Predication-driven parsing 56
2.7.3 Link Grammars 57
2.7.4 Unclassified 58
2.8 Commercialized checkers 58
2.8.1 Techniques used 59
2.8.2 Published and evaluated performances 59
2.9 Summary and conclusion 61

3 Theory 65
3.1 Constraint relaxation 65
3.1.1 Constraint definition 65
3.1.2 Constraint typology 66
3.1.3 Constraint violation 68
3.1.4 Prerequisites for constraint relaxation 68
3.1.5 Error diagnosis by constraint relaxation 69
3.1.6 Algorithm 70
3.1.7 Deciding which constraints to relax 71
3.1.8 Advantages 71
3.1.9 Problematic issues 72
3.1.10 Variants 75
3.1.11 Remarks 78
3.2 Phonological reinterpretation 78
3.2.1 Definition 79
3.2.2 Errors treated 79
3.2.3 Prerequisites 81
3.2.4 Algorithm 82
3.2.5 Advantages 84
3.2.6 Problematic issues 85
3.2.7 Variant 87
3.2.8 Systems using elements of phonological reinterpretation 89
3.2.9 Remarks 90
3.3 Chunk Reinterpretation 90
3.3.1 Definition of ‘chunk’ 91
3.3.2 Chunk reinterpretation 92
3.3.3 Errors to be treated 92

3.3.4 Prerequisites 94
3.3.5 Algorithm 94
3.3.6 Rule set 96
3.3.7 Advantages 98
3.3.8 Problematic issues 98
3.3.9 Variant 99
3.3.10 Remarks 99
3.4 Scoring 100
3.4.1 Syntactic likelihood 101
3.4.2 Detected error frequencies 102
3.4.3 Word frequencies 103
3.4.4 Combining the scores 106
3.5 Combining techniques 106
3.5.1 Independent approach 107
3.5.2 Cascade approach 108
3.5.3 Heuristic approach 109
3.5.4 Remarks 109
3.6 Conclusion 109

4 Implementation phase 113
4.1 Description of the Fips parser 113
4.1.1 Linguistic theory 114
4.1.2 Parsing algorithm 115
4.1.3 Implementation language 115
4.1.4 Advantages and disadvantages 116
4.2 Errors to be diagnosed 117
4.2.1 The FRIDA corpus 117
4.2.2 Selecting error categories 119
4.2.3 Error categories and examples 121
4.2.4 Error coding data structure 124
4.3 Implementation 127
4.3.1 Constraint relaxation 127
4.3.2 Phonological reinterpretation 137
4.3.3 Chunk reinterpretation 146
4.3.4 Analyses selection mechanism 148
4.3.5 Choice of technique combination 150
4.4 Conclusion 152

5 Results 155
5.1 Description of the test corpora 156
5.1.1 ‘Linguist’ sentences 156
5.1.2 Simplified sentences 157
5.1.3 Authentic sentences 157

5.1.4 Use of the different corpora 159
5.2 Evaluation 159
5.2.1 Quantitative results 160
5.2.2 Qualitative results 180
5.2.3 Efficiency 185
5.3 Comparison with another grammar checker 187
5.3.1 Comparison 188
5.3.2 Specifics of each grammar checker 191
5.3.3 Overflagging 192
5.3.4 Remarks 193
5.4 Conclusion 194

6 Conclusion 197
6.1 Contributions to the research domains 197
6.1.1 Usable and competitive system 198
6.1.2 Use of innovative techniques 199
6.1.3 Adapting an existing syntactic parser 201
6.2 Combination of several techniques 201
6.3 Limitations of the work accomplished 202
6.4 Further research 204
6.5 Final remark 207

A List of Abbreviations 209

B Sample test corpora 213
B.1 'Linguist' sentences 213
B.2 Simplified sentences 214
B.3 Authentic sentences 216
B.3.1 FRIDA data 216
B.3.2 FreeText data 217

Bibliography 223

List of Tables

2.1 Comparison between Le Correcteur and Antidote . . . 60
2.2 Comparison of three grammar checkers 61

3.1 Combining techniques 107
3.2 Comparison of the error diagnosis techniques 110

4.1 Selected error categories 122
4.2 Error categories and examples 125

5.1 Linguist sentences with auxiliary errors 161
5.2 Simplified sentences with auxiliary errors 161
5.3 Linguist sentences with class errors 162
5.4 Simplified sentences with adjective complementation errors 163
5.5 Linguist sentences with verb complementation errors . 165
5.6 Simplified sentences with verb complementation errors 165
5.7 CPV error breakdown on simplified sentences 167
5.8 Authentic sentences with euphony errors 167
5.9 Simplified sentences with gender errors 168
5.10 Simplified sentences with homonymy errors 170
5.11 Linguist sentences with number errors 171
5.12 Simplified sentences with number errors 172
5.13 Simplified sentences with negation errors 173
5.14 Simplified sentences with adjective order errors 174
5.15 Linguist sentences with adverb order errors 175
5.16 Authentic sentences with adverb order errors 175
5.17 Simplified sentences with person errors 176
5.18 Linguist sentences with voice errors 179
5.19 Simplified sentences with voice errors 179
5.20 Recapitulation of recall rates in % 180
5.21 Comparison of correct flags 188

Acknowledgments

My first thanks naturally go to Professor Eric Wehrli whom I have greatly appreciated as thesis supervisor. He allowed me to develop my work in whichever direction I chose, but he was always present when I wanted to discuss specific problems with him. Reading the first version of this dissertation by installments, he waited patiently, sometimes for several months, in between chapters. His comments were always valuable and allowed me to strengthen several aspects of my research.

I also want to thank Professor Eddy Roulet, president of the jury. His presence and his questions about the progress of my dissertation encouraged me at times when the end still seemed very far away.

I am thankful to Professor Thierry Chanier and to Professor John Nerbonne who kindly accepted to become members of my jury.

Many thanks go to all the members of the FreeText project, as most of the development of the grammar checker discussed in this dissertation occurred during the project's life time. Special thanks are due to Keith Potter who did most of the implementation work for the constraint relaxation technique, to the CECL team for the gathering, the keying-in, the error tagging, and the analyses of learner corpora, and to Marie-Josée Hamel with whom I wrote my first academic paper five years ago.

Current and past members of the LATL and of the Department of Linguistics must also be thanked for the quality of the working environment they helped provide. Eva Capitão merits special thanks for the numerous tea breaks and discussions held in her office, as well as her availability, her capacity to smooth all administrative hurdles, her permanent smile, and her excellent baking.

My current office mates, Catherine Walther Green and Sébastien L'haire, deserve a special mention. I learned a great deal from Catherine's numerous revisions and proof-reading of documents and papers I wrote over the past five years. Always ready to help, Sébastien was there whenever something did not work on my computer or when a new piece of software needed to be installed. Despite my moods, foul language, and legendary bad temper, they have both managed, through good humor and mutual teasing, to keep an overall cheerful and convivial atmosphere in what could have become a smelly hell.

Köszönöm szépen to my parents-in-law who, in March and April 2003, have taken took care of their grandson many afternoons in order to give me the necessary time and peace of mind to finish writing this dissertation. Without their help, I would not have been able to complete my work in this timeframe.

My deepest gratitude goes to my parents, Christiane and Thierry Vandeventer, who encouraged me to pursue my studies each time I contemplated taking another path. Their support brought me to where I am now. They must also be thanked for their share of child-caring during the last writing stages of this dissertation. I am sure that they enjoyed it more than the hours they devoted to proof-reading this dissertation, noticing many typos, errors, and strange sentence structures. For this unrewarding task, I thank them profusely. Their work was much more valuable than that of any grammar checker! Any errors that remain in this dissertation are my sole responsibility, but I can assure you that there are far fewer now than there used to be.

I cannot find appropriate words to thank my husband, Alexandre Faltin. Along with everything else, I am grateful for his constant support. When I was rushing to put the final touches to my dissertation, he transferred our son from day-mom to grandparents during his lunch breaks, and fetched him in the evenings, more often than his share. He did everything he could to make sure I had the time I needed to finish my work. I only hope to be able to do the same for him someday soon.

Last, but certainly not least, I want to thank my son, Raphaël, for being such a good baby and for reminding me that there is life after, and during, dissertation writing.

Chapter 1

Introduction

Spoken or written, language surrounds us and is our most common means of communication with other people. Nevertheless, language is at the same time a barrier to communication because of the great diversity of languages and because each individual speaks only a few of them. In order to overcome this barrier and be able to communicate with speakers of other languages, many people learn one or several languages besides their native language.

Second language acquisition is a difficult task, especially for adults. There are numerous methods to acquire a new language and all of them require some form of feedback, which can be described as a reaction to what has been said or written[1]. This feedback most often comes from other human beings with whom the language learner is interacting. There are, however, other means to receive feedback. One is the use of computer assisted language learning (CALL) software. Typically CALL software contains exercises for language learners. Their responses to the exercises are analyzed by the system which provides some form of feedback.

Some types of exercises are very easy to correct because the number of possible answers is very limited, such as with multiple choice questions and gap filling exercises. Simple methods can then be used to provide a simple feedback to the users. Whenever the range of possible answers is large, or even infinite, specialized tools are needed. In the case of exercises requiring users to produce sentences in the language they are learning, natural language processing (NLP) tools are necessary to analyze the answer and produce intelligent feedback.

Definitions of natural language processing (NLP) or computational linguistics are numerous in literature devoted to the domain. To cite

[1]See (Ellis 1997, p. 43-50) on the importance of both input and output for language acquisition.

only two, Tennant states that "research in natural language processing is concerned with making computers capable of using natural languages" (Tennant 1981, p. 2), while Grishman writes that "computational linguistics is the study of computer systems for understanding and generating natural language" (Grishman 1986, p. 4). NLP is a large field in which every practitioner sees something slightly different. Domains of NLP include natural language interface, machine translation, text analysis, information extraction, speech understanding and synthesizing, and some areas of CALL, to name only a few.

The work presented in this dissertation concentrates on NLP tools designed for the diagnosis of grammar errors produced by language learners in the context of a CALL system.

The remainder of this chapter is structured as follows. Section 1.1 sets down the main problem discussed in this work. Section 1.2 states the theoretical as well as practical objectives of the work. Section 1.3 explains the methodological approach chosen. Section 1.4 gives a brief overview of the research project in which the work giving rise to this dissertation has been carried out.

1.1 The problem

Errors can be detected only because there is a norm of what is considered "correct" language. It is relatively easy for native speakers to notice errors in language productions because they know what the norm is, even if unconsciously. This gives them the possibility to discover discrepancies between the norm and actual productions. Chomsky (1965) distinguishes between the competence of speakers, their unconscious linguistic knowledge of the language, and their performance, their actual use of the language. Because of performance issues, actual language productions do not all respect the norm. "A record of natural speech will show numerous false starts, deviations from rules, change of plan in mid-course, and so on" (Chomsky 1965, p. 4). It is difficult to ensure that all of one's own productions respect the norm, and errors abound in the language that surrounds us. Spoken and written productions give rise to different types of mistakes, but written sentences are less prone to mistakes because the writing process usually includes some revision time when many mistakes can be detected and corrected.[2] Mistakes of all kinds still do appear in written texts, "such

[2]Discussion will be limited to written language in this dissertation for reasons described in section 1.2.3. We will not, therefore, go further into the differences between spoken and written language from here on.

as misspelled words, missing words, poor syntactic constructions, unclear or ambiguous interpretation, missing crucial punctuation, etc." (Granger 1983, p. 188). Typed text introduces another layer of possible mistakes, linked to the typing process, commonly referred to as typos. Even published texts, in newspapers, journals, and books, contain mistakes now and then.

1.1.1 Treatment of ill-formed input

The human mind is very flexible and can adapt quite easily to the errors it encounters, sometimes not even noticing them consciously. Noticing mistakes does not necessarily prevent us from interpreting sentences, giving them one or several grammatical structures and interpretations, and forming hypotheses as to the source of those mistakes. These hypotheses may then be used to provide feedback to language learners. Unfortunately, computer programs are much less flexible and run into difficulties when their input differs from the norm they are expecting. NLP applications generally assume that the input they will receive is error-free. Although this is a practical assumption which helps to find solutions and to limit the complexity of an already hard problem, it is not realistic. An important criterion in the evaluation of NLP systems is their robustness. "*Robustness* refers to the quality and capability of a program. When a program is said to be robust, it works well and doesn't fall apart when some unusual condition occurs" (Freedman 1989, p. 597, emphasis in original). The minimum requirement for a robust system is to not crash when encountering an unexpected input and not terminate instantaneously. Systems that fold down at the first discrepancy from the norm do not answer this criterion. To reach a certain level of robustness, NLP systems must find ways to handle ill-formedness, whether it be skipping it or providing some form of output for this unexpected input.

The treatment of ill-formed input varies widely among systems depending partly on their end usage. Let us take the case of the appearance in the input text of an NLP system of a word which does not belong to the lexicon of this system, an unknown word.[3] A text interpretation system, such as NOMAD (Granger 1983), will try to derive the meaning of the unknown word from its context. A speech synthesizer will rather include heuristics to pronounce unknown words based on their orthographic representation, and possibly on their expected lexical category (Gaudinat and Goldman 1998). A translation system

[3]For the sake of the example, let us assume that this word does not start with a capital letter and will not be considered to be a proper noun.

might render the erroneous word "as is" in the translated text, while a syntactic analyzer will try to guess the lexical category of the word based on its morphological features and its syntactic context (Erbach 1990).

A spell checking system, on the other hand, is specifically designed to handle such a case. The unknown word is considered to be the result of an error on the part of the writer. The checker usually highlights the word and tries to give possible, correctly spelled, alternative words. Spell checkers are part of the group of writing help tools, whose role, far from ignoring erroneous input, is to detect ill-formedness in their input. Commercially available writing help tools include most commonly spell, grammar, and style checkers for native speakers and are often included in text processing software.

1.1.2 The CALL context

Within the CALL context, ill-formed input takes another dimension. Apart for spelling errors which are often due to a lack of competence in this specific area, mistakes made by adult native speakers are due in a very large part to performance issues. Language learners, however, make mistakes for both performance and competence reasons. Thus, input provided by language learners contains on average more errors than input produced by native speakers. Moreover, the types of errors committed by these two groups are not similar (see (Granger and Meunier 1984, p. 80)).

Types of errors

The types of errors made by language learners include the types made by native speakers and add to them all the errors due to competence issues. For example, as can be seen in (1) below, language learners create words which do not exist in the language because they use erroneous derivation paths (1a). They mistake homophones for one another more often (1b). They do not necessarily respect word order constraints (1c). They do not always know the gender of nouns (1d) or the auxiliary to be used with a given verb (1e). They do not respect the rules of euphony (1f). And they misuse punctuation (1g), to name only a few types of mistakes seen in language learners' texts.

(1)a. *[4] *nonpossible*
notpossible

[4]The star * indicates an erroneous word or phrase.

b. *reine* instead of *règne*
queen instead of reign

c. * *Il jamais n'a vu.*
He never negation has seen.

d. * *la soleil*
the-feminine sun-masculine

e. * *J'ai né.*
I have born.

f. * *ce endroit*
this place

g. * *Il mange, et il boit.*
He eats, and he drinks.

NLP for language learners

In this work, we will make a distinction between non-native speakers and language learners. However, let us start with the definition of a native speaker given by Akmajian, Demers, and Harnish (1984). A native speaker is "a person who speaks a language fluently, typically because that person has been brought up speaking that language as a child" (Akmajian, Demers, and Harnish 1984, p. 525). A non-native speaker, then, is someone who does not speak with the fluency of a native speaker. We will define language learners as non-native speakers who actively seek to improve their competence of the language.

Although one cannot exclude language learners using any type of NLP tool, such as a natural language interface to question a database, the NLP systems they are most likely to use, and provide their own input to, are writing help tools, such as spell, grammar, and style checkers, and obviously language learning softwares. Very often, language learners use writing help tools designed for native speakers because there is none designed for language learners easily available. Checkers designed for native speakers are often not sufficient, as we have just seen that language learners make more and different kinds of mistakes than native speakers. This is confirmed by Jacobs and Rodgers (1999) who write that "it should be noted that since French grammar checkers are aimed at native French speakers, their usefulness as learning and teaching tools may conceivably differ with native speakers of other languages" (Jacobs and Rodgers 1999, p. 522). Moreover, language learners want informative feedback on their productions. A 'good/bad' answer is not sufficient in the context of exercises. "In addition, they demand a greater flexibility of the programs which would make it possible for CALLware to accept more than one answer in a given situation and context" (Buchholz 1992, p. 135-136). There is therefore a need to create and make available checker tools designed specifically for language learners.

Diagnosis vs. correction

Diagnosis and correction are two distinct notions. The Concise Oxford Dictionary states that diagnosis is "the identification of the cause of a mechanical fault etc." (Allen 1990, p. 321) while correction is "a thing substituted for what is wrong" (Allen 1990, p. 258). To propose a diagnosis, one must state whether there are mistakes in a given sentence, give their locations, and indicate the error types to which they belong. Establishing a diagnosis does not involve giving correct alternatives for the errors detected. Providing correct alternatives is part of the correction process. Naturally, the correction process must start with a diagnosis phase, otherwise it would be unable to know what needs correction. Therefore, correction includes diagnosis and completes it with possible solutions for the errors.

Example (1e) is given again below as (2a). (2b) is a possible diagnosis for it and (2c) a correction.

(2)a. * *J'ai né.*
I have born.

b. Auxiliary *avoir* (have) cannot be employed with verb *naître* (to be born).

c. The correct sentence is *Je suis né.* (I am born.)

The distinction between diagnosis and correction is important both for checker tools and for the CALL context. Very often, commercial checkers propose not only an error diagnosis, but also a correction. Spell checkers, in particular, offer a short list of possible alternative words to replace the unknown, and supposedly erroneous, one. By providing not one, but several possible solutions, and finally letting the users choose whether or not to correct their sentence, the checkers are less likely to replace an error by another one. All commercial spelling checkers, to our knowledge, not only establish a diagnosis on the input, but also propose a correction. Calling them 'checker' is therefore slightly misleading as they do much more than check. The terms 'check' and 'checker' will be used in the rest of this work to refer to the diagnosing act and to the diagnosis, unless stated otherwise.

While native speakers appreciate having corrections proposed to them, allowing them to correct their texts faster than if they had only diagnosis results available, the case is slightly different for language learners. Second language learners not only want to correct their mistakes on the short term, but they also want to improve their language skills in order not to do the same mistakes over and over again. Language learners produce both performance and competence errors. When faced with performance errors, they are essentially in the same position as native speakers: an indication of the location of an error and of its type is largely enough for them to correct the error. Naturally, a proposed correction for that error would facilitate the correction work. It is however difficult to always distinguish between performance and competence errors with language learners, as attested by Ellis (Ellis 1997, p. 17-18), especially as two learners can make the same mistakes for two very different reasons. Moreover, "some research shows that direct correction is not particularly effective" (Krashen 1984, p. 118) and "the 'self-correction' method is found to be more desirable for learning" (Yang and Akahori 1999, p. 73). Therefore, only a diagnosis will be provided, letting the learners find out the correct solution for themselves. "In a CALL program, error messages should point the learner in the right direction for correction rather than supplying the correct version directly" (Tschichold 1999, p. 215–216). One should keep in mind that CALL caters to language learners rather than simply to non-native speakers. If one were to design tools for this latter kind of users, then having correction available would certainly be a necessary feature for marketability.

Overflagging

A good diagnosis system "must not only accurately identify the nature of an error but must also avoid falsely identifying mistakes in correct words" (Burston 1998). Overflagging, which occurs when the system indicates an error although the input is correct, is a recurrent problem of all diagnosis and correction tools (see (Tschichold 1999, p.213)). An occurrence of overflagging is also called an overdetection, a false positive, or an erroneously detected error. Overflagging is damaging to all systems. Users do not like it as it takes their time unnecessarily in order to verify each overflagging occurrence to see whether there is indeed an error or not. Overflagging also shows some of the limits of the systems. A system that overflags often is not as reliable as one which does not. "More worrisome, from a pedagogical point of view, is the fact that calling attention to forms that are correct as possibly incorrect may confuse the learner unnecessarily" (Hull, Ball, Fox, Levin, and McCutchen 1987, p. 106). Native speakers, and proficient non-native speakers, can deal with overflagging because their competence of the language is good enough to decide for themselves whether the sequence that has been erroneously flagged is in need of repair or not. On the other hand, language learners do not possess this knowledge and overflagging can be highly detrimental to them. In the best case, they will eventually figure out that their text was correct, contrary to what the system showed. It might take them some time to reach this conclusion, however. In the worst case, they will amend their sentence to try to better it. In this case, not only will they risk introducing mistakes into a sentence which was correct in the first place, but they might also believe that the correct sentence structure they used is not part of the language they are learning. They might, in fact, be mis-lead by the diagnosis system into mislearning, or integrating into their grammar of the target language rules that do not belong to it. Obviously, misleading the learners into learning wrongly is to be avoided. It is therefore of the utmost importance to have a rate of overflagging as low as possible. A low rate of overflagging reduces the risk of mislearning, increases the usability of the system, and gives a better impression of the system to the users.

1.2 Objectives

The objectives of the work undertaken contain a theoretical component as well as a practical one. Both revolve around NLP tools for CALL purposes and more precisely around checker tools for written input. We will start this section by describing in more details what these checker tools are and by specifying the range of tools we will be concerned with. We will then discuss the theoretical and practical objectives that gave rise to the present work.

1.2.1 Levels of checking

To build a comprehensive checking system, several levels of diagnosis are necessary. To be complete, one should probably consider four diagnosis levels: spelling, grammar, semantics, and possibly style. These levels interconnect to some extent but, for simplification's sake and for the time being, we will consider that they are independent. To our knowledge, there is not, to this day, a single checking tool that encompasses all these levels.

The role of a spell checker is to verify that the words appearing in a document all belong to the lexicon of the system. This is the simplest level of checking and was included quite early in text processing software (see (Peterson 1980)). Moreover, spell checkers do more than just check the entry: they also propose alternatives for the words they cannot find in their lexicons and are therefore correctors rather than true checkers.

The second level of diagnosis is grammar. It appeared later in text processing software. The goal of this tool is to inform its users as to the grammaticality of the sentences they are writing. Most commercial tools include some type of correction as well. They do not, however, treat all sentence constructions and are not able to detect some of the errors committed by the users.

The next diagnosis level is semantics. Semantic checking should be concerned with the coherence of meaning of the words and sentences used within a text. It should ensure that the text does not contradict itself, that tense concord is respected, that pronouns correspond to their referents, among other things. Full semantic checking requires the system to possess a knowledge of the world, which is difficult or even impossible to implement on unrestrained domains. Understandably, semantic diagnosis is not, to our knowledge, available in any large coverage system. Nevertheless, one can adopt a more modest approach than full semantic checking and provide some kind of coherence checking. Although not covering all the points noted above, coherence checking with limited means remains interesting as long as it provides new information on the input under scrutiny. In the CALL context, coherence diagnosis can indicate whether the learners' input is in relation with what was asked of them.

Finally, many commercial checking systems nowadays combine a style checker with their grammar checker (see Cordial 7 (Synapse) and the grammar checker included with the French version of Microsoft® Word 2000). A style checker is not supposed to inform the users of sentences that are not grammatically correct, but simply of a style that is not adequate for the kind of text they are writing. Style checkers, for instance, inform the users of the presence of too many passive sentences in a text, which is deemed to be poor writing. Style is by nature subjective and poor style does not prevent a sentence from being acceptable on a grammatical basis. Thus, style stands somewhat apart form the other domains of diagnosis and will not be considered further in this work.

The main purpose of this work is to build a grammar checker, that is, a tool capable of indicating whether any given input sentence is grammatical or not according to the norm of the language. Although it is obvious that the spelling and semantic levels of checking are important, we will focus here on the grammatical level of checking only. Error types that should be diagnosed by a grammar checker include agreement, word order, the choice of auxiliary, and euphony, to mention only errors in example (1). A more complete list is given in chapter 4, section 4.2.

1.2.2 Theoretical objectives

On the theoretical side of the work, the objectives include an investigation of existing grammar checkers, both prototypes and commercial systems, both for native speakers and for language learners. Through this study, we hope to learn what has been done in the past and what is currently done concerning grammar checking. We will, whenever possible, examine the diagnosis algorithms of grammar diagnosis systems and their implementation. We will then evaluate the different diagnosis techniques that have been used and their potential.

Starting from the knowledge acquired through this study, we will concentrate our efforts on three different techniques: constraint relaxation, phonological reinterpretation and reinterpretation of a sequence of partial analyses (chunks) resulting from an incomplete parse. We will propose algorithms to use these three techniques individually, trying to figure out their limits and their advantages. We will then investigate how these three techniques, and any technique by extension, can be combined in order to increase their individual effectiveness.

1.2.3 Practical objectives

On the practical side of the problem, we want to create an effective grammar checker for the CALL context, resulting from the theoretical considerations mentioned above. This diagnosis tool should be able to diagnose grammar errors in authentic texts produced by language learners.

Let us pause a moment and limit somewhat the extent of the grammar checker that we want to ceate in order to exemplify the theoretical part of the work. First of all, it should be mentioned that we are concerned with written language exclusively. The rationale behind this restriction of the production medium is threefold. Firstly, voice recognizers do not have a good enough performance to recognize accurately oral productions spoken by native speakers, let alone language learners, without extensive training. Secondly, spoken language is much less formal than written language and is also more open to regional variations in what is acceptable or not. Thirdly, and most importantly, spoken language is usually unprepared and spontaneous, and thus open to interruptions, false starts, restarts, and hesitations. Written language, especially in French thanks to the *Académie française*, is more highly homogeneous and it is, therefore, easier to determine what is part of the language and what is not, what is grammatical and what is not.

Using a grammar checker within a CALL context also implies some further restrictions. For didactic reasons, as mentioned above, a checker needs not, and should not, be a corrector. Indeed, learning is more efficient when it is an active process rather than a passive one. Having to find for oneself the correction of one's errors leads to better learning than finding the correction without any effort. Therefore, the grammar checker that we propose here is not coupled with a correction device, as already mentioned. It must, however, provide sufficient information for the CALL software to indicate to the users the types and locations of the errors detected. The CALL system is then able to direct the users towards relevant parts of the tutorials or of a reference grammar in order to provide the users all the necessary clues to facilitate their search for the correct formulation. The exact form the feedback should take is not of primary concern in a grammar checker to be inserted within a CALL environment. Questions of form and of display belong to the didactic approach of the CALL system. We are, however, concerned with the kind of information which has to be transmitted from the diagnosis tool to the CALL software.

An advantage brought about by working in a CALL environment is that we are able to make use of the context to help the diagnosis process. Indeed, in a CALL software, what can be written as part of exercises is much more restricted than when using a text processor. The format of the exercises limits the type of answers that can be given. Moreover, the questions can guide the diagnosis system with indications of what to expect in the answer.

Finally, there are some characteristics of the targeted users of this grammar error diagnosis system which need to be taken into account. The envisaged users are intermediate to advanced learners of French. They already have a good command of the language that they can have acquired within or outside formal classroom settings. For this reason, it is nearly impossible to determine precisely the range of constructions that they master or will attempt to use and the coverage of their target language lexicon. This forces the diagnosis system to have a large coverage itself, in order to recognize all instances of what the learners are trying to express. One cannot rely on a given language method and assume that the users will know all and only the structures and vocabulary taught in that method up to a given chapter. The diagnosis system must thus be as general as possible.

Another reason for a general approach to the system is that we have no prior knowledge of the mother tongues of the users. Transfer errors from the first to the second language are well attested in literature (see (Tschumi 1994), among others, for interference errors in verb tenses in English writing of French speakers). When designing a diagnosis system for learners of a given first language, one can incorporate the information on typical transfer errors for that type of users (see for example (Weischedel and Black 1980, p. 99) and the works the *Laboratoire de traitement du langage et de la parole, Université de Neuchâtel* (Cornu 1994, Tschumi, Bodmer, Cornu, Grosjean,

Grosjean, Kübler, and Tschichold 1994)). This option is not available in the present case, as our target users speak a variety of mother tongues. This forces us to keep a distance, not to base our work on transfer errors, and to realize a more general diagnosis.

To sum up, the objectives contain a theoretical part which is the study of existing grammar checkers, both at the prototype and the commercial stage, as well as the discussion of selected algorithms that can be used for grammar checking, either in isolation or in combination with one another. The practical part of this work involves actually implementing these algorithms in order to create a grammar error diagnosis tool for the CALL context.

1.3 Methodological approach

Work that has led to this dissertation is anchored within a practical project.[5] The needs and scope of this project had to be taken into account in the realization of the grammar error diagnosis tool, which had a number of implications concerning the methodological approach. The duration of the project was set in advance and the diagnosis system had to be completed within that time frame. For this reason, it was very important to make use of all possible resources available, such as the Fips parser (see (Laenzlinger and Wehrli 1991, Wehrli 1997) for a description of the parser), and not to try to build everything from scratch again. Requirements for the completion of the project were also taken into account, in particular concerning the errors to be treated. As it is realistically impossible to diagnose all kinds of errors, a selection of errors was made conforming primarily to the needs of the project. Finally, what we were building had to work at the end of the project, however we managed to achieve this. This fact helped us stretch the theory, invent new diagnosis techniques and combine them in order to have a working system at the completion of the project.

1.3.1 Selection of errors

Thinking utopic that a single error diagnosis system could be able to diagnose all the kinds of errors appearing in a text, even limited to grammar errors, a selection of the types of errors to be treated by the diagnosis system is an important point that shapes the whole system.

Three questions can be asked and we will try to answer them in order.

1. Which error types can be diagnosed?
2. Which error types should be diagnosed from a didactic standpoint?

[5]The FreeText project aims at creating a complete CALL software for intermediate to advanced learners of French, including a diagnosis system. For a general description of the project, see section 1.4.

3. What are the error types actually found in learners' texts?

To know what error types can be diagnosed, we can study the literature, see what other systems are able to do and try to do at least as much. We can also draw on our knowledge of the tools available and infer what can be transformed and how it can be transformed, and thus what can be diagnosed. But this latter solution has the disadvantage of probably presenting us with only the error types that are easy to diagnose, because they are those that first come to mind. Some complex but more important errors may be missed that way.

Didactic specialists will be able to provide a list of error types or points that they desire to see covered by the diagnosis system. This list is usually drawn from their experience of the error types found in learners' productions, from their own views on teaching, and from the content of their didactic materials. Language teachers want to be able to consider whole texts in their materials and therefore have a need for diagnosis tools that go beyond the sentence level to reach into semantics and pragmatics, without forgetting to have as large a coverage as possible.

It is also important to know what errors are actually made by learners and teachers' intuitions are not necessarily enough in that matter. The best way to know this is to collect learner corpora and to analyze them. This includes tagging corpora by hand for as many types of errors as possible. It is then possible to analyze the errors statistically. Error frequency is one of the simplest analyses available, but there are countless more complex ones, such as knowing that in 95.5 % of the data in the corpus, when a number agreement error occurs between a subject noun phrase and a verb, it is the verb which is not pluralized properly.[6]

The didactic needs and the knowledge of errors actually found in learners' texts stretch the demands on the diagnosis system and forces NLP researchers to enlarge the number of errors that they had thought interesting to diagnose, as well as pushing them to find ways to diagnose these errors and therefore to expand the theoretical limits of error diagnosis.

The selection process for the errors to be diagnosed by the system described in this dissertation tries to put in balance what can be detected, what is desired didactically and what is actually found in learner corpora. The final selection, which is described in more detail later (chapter 4, section 4.2) depends partly on the grammatical points stressed within the tutorials of the whole CALL software and on the error types and frequencies found in learner corpora.

[6]Data from Sylviane Granger, personal communication, Manchester, October 2000.

1.3.2 Reuse and adaptation of a parser

One of the key idea of the chosen methodological approach is to use, reuse, and adapt whatever tool is available rather than to recreate it anew from scratch. Very often, project designers must make a choice between, on the one hand, creating a prototype from nothing, knowing that they will not be able to construct a complete working system but hoping that their prototype will be large enough to demonstrate that their hypotheses are working; or, on the other hand, reusing, modifying and adapting existing tools in order to create a usable system, at the price of adapting their hypotheses to the available materials. We now look at the rationale underlying the reuse of a syntactic parser in the present case, then quickly sketch the kinds of adaptations needed, and finally evoke some of the problems or disadvantages associated with the reuse of particular NLP tools.

Rationale

By reusing tools, and in particular a syntactic parser, one saves time and money. The creation of a syntactic parser is a very time consuming task which therefore also requires some amount of funding to employ the necessary people. "It can take years to develop an NLP tool that provides accurate grammaticality judgments and it therefore seems more efficient not to write a parser from scratch but rather to use an existing one" (Schulze and Hamel 2000, p. 86). In a project like the one linked to this dissertation, the time and resources needed to create a whole parser from nothing were not available. If we had not had a parser already available, the project would not have been born.

Apart from the gain in time and money, reuse of a syntactic parser brings us robustness, reliability and coverage. Each of those necessary features is not easy to achieve but are the elements that differentiate a complete system from its prototype. In the present case, the three elements are of particular importance. Robustness is crucial because language learners tend to produce ungrammatical sentences that the parser will be given as input. If the parser were to crash at any deviation from the norm of grammatical language, the system would be useless for this application. Reliability is important because, while native speakers can have a critical view on the output of the NLP tool thanks to their knowledge of the language, this is much more difficult for language learners: their knowledge of the target language leaves them no other possibility than to trust the feedback they receive from the NLP tool. Finally, coverage, including coverage of the lexicon, is absolutely necessary if one wants to cater to the needs of intermediate to advanced learners. This type of learners may have acquired their knowledge of the language in very diverse situations, they might use sentence structures and vocabulary that are perfectly correct but not often found in language courses. It would be damaging to force those learners to conform to what is taught in a given textbook, hence the importance of a large coverage.

Finally, another advantage of reusing a syntactic parser is that it can serve first as a recognizer, thus indicating whether a given sentence is grammatical, or not. This is a great help in the case at hand. We are trying to detect ungrammaticalities in the input produced by language learners. Ideally, a first pass through the parser, serving as a recognizer, will already tell us whether the sentence is grammatical and needs no other treatment or if it is ungrammatical and must undergo a diagnosis procedure.

Adaptations

The first type of adaptation the parser has to undergo concerns the particular context of use. The Fips parser is a generic parser in the sense that it provides syntactic analyses of its input which can then be employed in a rather wide range of situations such as in a speech synthesizer, and an automated translation system. Although the Fips parser was not planned for any particular end application, it was assumed that its input would be grammatical. This is a way to slightly reduce the difficulty and the complexity of the parsing problem. This is not to say that Fips is not robust, quite the contrary: Fips provides partial analyses for its input when it cannot find a syntactic structure covering it completely. Fips was nevertheless implemented with the view that its input would be texts written with care by native speakers, containing almost no mistakes. A consequence of this is that it was never really tested before with ungrammatical input. Unfortunately, this exercise revealed that the Fips parser provides full analyses for many ungrammatical sentences and thus considers them to be part of the language. A first necessary adaptation is therefore to tighten the parser to prevent it from recognizing ungrammatical sentences as grammatical.

At the same time, it is also important to improve the coverage of the parser. As a general rule, coverage can always be extended as there is no single tool, to our knowledge, that covers a natural language completely. In the present case, one has to ensure that the parser is able to parse correctly the different text types and sentence structures found in the authentic documents of the tutorials, as well as in the exercises, the reference grammar and all other kinds of materials included in the CALL software developed for the FreeText project.

The second type of adaptation of the parser is its transformation into a syntactic error diagnosis system able to diagnose a set of error types defined in conjunction with the needs of the CALL software of which it will be part. For the diagnosis of each error type, there should be a phase devoted to the study of the parser in order to understand how the particular problematic area is analyzed. A good knowledge of how the parser works is absolutely fundamental before anyone can make suggestions of modifications to diagnose a particular error type. Once a proposal of modification is made, especially if it concerns many structures and/or areas of the code of the parser, there should be a test on a sample, in order to see whether the preliminary results look encouraging. If so, the modification can then be implemented full scale. Otherwise, one should revise the proposal and start again.

Disadvantages

Obviously, reusing and adapting existing NLP tools has also its share of disadvantages and problems. Because one is not starting from scratch, it may be necessary to adapt one's hypotheses in some ways to the tools available and to take into consideration their architecture to see where modifications can take place without side effects. Thus, there are some constraints on the kinds of diagnosis techniques that one is able to use with the Fips parser. These constraints depend on the architecture of the parser and its working algorithms. Techniques that can be implemented in the Fips parser are used for that very reason, while others, also very interesting on a theoretical basis, are rejected because they are not compatible with the parser we are reusing. A diagnosis technique that fits the Fips parser well is constraint relaxation (which is used in numerous diagnosis systems, even if under somewhat different names, see (Weischedel and Black 1980) and (Heinecke, Kunze, Menzel, and Schröder 1998)). One can also benefit from Fips's work as a recognizer providing partial analyses to reinterpret those partial analysis to diagnose why structures where not able to combine. One gain of reusing a parser is to be able to provide a syntactic structure of the input besides a diagnosis. This already dismisses diagnosis techniques which do not rely on parsing such as pattern matching and the use of automata to look for specific errors (see (Tschumi et al. 1994) and (Kübler and Cornu 1994)). Although error grammars and error rules diagnosis techniques are in fact able to parse their input (see (Murphy, Krüger, and Grieszl 1998) and (Schneider and McCoy 1998)), such techniques are not usable with Fips because they require a rule set completely separated from the parsing algorithm and this is not the case with Fips.

Finally, as the combination of techniques that are used in the end is very dependent from the parser we are adapting, the whole combination is not easily transferable "as is" on another parser. Nevertheless, individual techniques should be transferable, and the theoretical hypotheses advanced in this dissertation should be generic enough for reuse elsewhere.

1.3.3 Combining diagnosis techniques

Within the error diagnosis system, several techniques are used concurrently. This can increase the number of error types which are diagnosed, decrease the number of overflagging, but also create a new challenge.

Increasing the number of error types diagnosed

Each diagnosis technique is able to diagnose a specific range of error types. Inversely, each technique also has a number of error types that it is not able to treat because the technique is not adapted for this purpose or because it would be too costly to implement it for a particular error type. By using several techniques, one can try to counterbalance the fact that some error types are not treated by a given technique. If several diagnosis techniques work side by side, we can hope that their respective set of treated errors are not identical, that they are even nearly disjoint, and that one can thus correct many more error types by using several techniques, each technique targeting a particular set of error types.

Decreasing the rate of overflagging

Because the sets of errors targeted by particular diagnosis techniques are not completely disjoint, using several diagnosis techniques should also help us to decrease the rate of overflagging. Let us consider the case where only one technique gives a diagnosis on an error type. If this technique indicates that an error is present, one must consider it so and warn the user, running the risk that this was an occurrence of overflagging. Now, if two techniques target the same error type, and if both independently indicate that a particular error is present, it is far less likely that they are both wrong. Thus, by having two, or more, techniques providing the same results, we reach a higher level of certitude that the error can actually be found in the text and that we are not overflagging the user.

The challenge

The results of two or more techniques do not necessarily coincide, however. This raises the problem of how to interpret divergent results and, in general, of how to combine in the most efficient way several techniques and their results. Several ideas come to mind. One could use all the techniques, one after the other or simultaneously, and compare their results, perhaps using a weighing system depending on the technique and the error type, as not all diagnosis techniques are as good with a specific error type. One could use only one technique at a time, but decide which one to use depending on the results of a first pass through the parser. Or one could start with a technique and, depending on intermediary results, change techniques midstream if one considers that another technique would be more appropriate for the task at hand. Solutions to these questions will be discussed, and hopefully answered, at the end of chapter 3, in section 3.5.

1.4 The FreeText project

A large amount of the work described in this dissertation and most of the implementation of the diagnosis system were made in the context of the FreeText project which is described briefly in this section.

The FreeText project is a collaboration between four partners located in Europe: the Department of Language Engineering of the University of Manchester Institute of Science and Technology (UMIST), United Kingdom; the Centre for English Corpus Linguistics (CECL), *Université Catholique de Louvain*, Belgium; Softissimo SA, France; and the Department of Linguistics, *Université de Genève*, Switzerland. It is funded under the auspices of the Fifth Framework Programme of the European Commission.[7] The aim of this three-year project, started in April 2000, is to create a computer assisted language learning software for French as a foreign language. The software targets users at the intermediate to advanced levels. It is innovating in that it uses authentic documents and a communicative approach to second language acquisition (SLA) as a basis for meaningful task-based activities. The content of the tutorials takes into account learner corpus analyses to tailor the exercises to the needs of the target users. NLP tools are used for error diagnosis in exercise answers and are moreover available to the users at all times for testing and illustration purposes. The error diagnosis system, which is made of a spell checker, a grammar checker and a 'semantic' checker, relies also on the error typologies and analyses extracted from the learner corpus.

Each partner in the FreeText project has its specific set of tasks. UMIST is responsible for the didactic content of the software. The CECL is in charge of everything relating to corpora and is also responsible for the validation stages of the project. Softissimo works on the interface and on the integration of all the parts into a coherent software. The Department of Linguistics of the *Université de Genève* develops the NLP tools necessary for the project.

This dissertation is concerned mainly with the conception and realization of the component of the diagnosis system treating grammar errors.

For a more detailed description of the FreeText project, see the project website:[8] http://www.latl.unige.ch/freetext/

[7]The FreeText project receives financial support from the European Commission in the IST programme of the 5th framework-programme, contract IST-1999-13093. The Swiss participation is funded by the Swiss Federal Office for Education and Science, contract 99.0049. The content of this document is the sole responsibility of its author and does not represent the opinion of the Community. The Community is not responsible for any use that might be made of the data appearing in this document. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The users thereof uses the information at their sole risk and liability.

[8]Last accessed on April 7, 2003.

1.5 Structure

This dissertation is structured as described in the following paragraphs.

Chapter 2 describes the state of the art in the domain of grammar error diagnosis. It starts with a brief history of the treatment of ill-formedness in NLP and CALL. An indication of what is done nowadays concerning spell and semantic checking follows. We then describe several techniques used for error diagnosis. These techniques include pattern matching, finite state automata, error grammars and constraint relaxation. Each technique is described in turn and examples of systems using them are given. A study of a few commercial tools is included, specifying the mother tongue of its target users, the techniques they use when known, and available performance results.

Chapter 3 contains the theoretical hypotheses of this dissertation. Three techniques are described, constraint relaxation, phonological reinterpretation, and chunk reinterpretation, including for each technique possible variants to the main theory and indicating how each technique differentiates itself from the others. Finally, we investigate four different hypotheses to use the above mentioned diagnosis techniques in combination and to manage the results coming from different sources. These hypotheses are a sequential ordering of the techniques, a parallel activation of them with competition of the results, a cascade hypothesis where one could switch technique mid-stream and finally a heuristic hypothesis where the choice of technique would be determined by the results of a first pass of the input through the parser.

Chapter 4 relates the actual implementation of the hypotheses described in the preceding chapter. It starts with a description of the syntactic parser which has been transformed into a diagnosis tool and continues by explaining the range of errors that were selected to be diagnosed by the system and their representation in the system. A report is then given of the implementation of the diagnosis techniques, indicating the variant used, the problems encountered and the limits of the present implementation. Finally, the type of combination of techniques implemented is detailed.

Results are given in chapter 5. It contains a description of the corpora which were used to test the system. We give quantitative results in the form of statistics, qualitative results based on human observation of the data, and comments on those two kinds of results. This is followed by a comparison of our system with a commercial grammar checker.

Chapter 6, the conclusion, highlights the contributions to the research domains in terms of the emergence of a competitive diagnosis system and of the use of some rarely employed diagnosis techniques, without forgetting the cooperation between several techniques. Some limitations to the work accomplished are detailed and, finally, some leads for further research are given.

Chapter 2

State of the Art

This chapter gives an overview of the state of the art in the treatment of ungrammatical sentences in natural language processing systems, covering roughly the 1980's and 1990's, extended by a couple of years in both directions. The first section reviews a short history of NLP for error diagnosis in CALL systems. Section 2 briefly discusses spell checkers today. As one considers that "the degree of intelligence that grammar checkers display varies depending on the techniques used for language analysis" (Chapelle 1989, p. 63), it seems important to discuss existing techniques. Thus, the following five sections explore diverse techniques for treating ungrammatical sentences. Each technique is described in detail and examples of actual systems using the technique are given. The described techniques include pattern matching, finite state automata, error grammars, constraint relaxation, and other techniques less well represented. Section 2.8 relates the information known about some commercialized grammar checkers. Finally, section 2.9 concludes this chapter.

2.1 A short history of NLP in CALL

This section presents a short history of NLP for the treatment of ill-formed input in CALL. The aim is not necessarily to be exhaustive, but to present a summary of what has been done and of the focus of attention in the 1980's and 1990's, extended by a couple of years in both directions. For a history of CALL itself, starting from the very beginning, the reader is referred to Chapelle (2001) and to the following web site,[1] http://www.history-of-call.org/ from which some of the information in this section was gathered. A interesting survey of intelligent CALL (ICALL), including NLP applications and covering the 1990's up to 2002, is given in (Gamper and Knapp 2002). For a very well documented review of NLP in CALL, for all kinds of

[1]The site was last accessed on April 14, 2003.

applications besides error diagnosis and correction, see (Nerbonne 2003).

2.1.1 The early 1980's

NLP practitioners very often assume, as a simplification, that their system's input is grammatical. Thus, early systems were less than informative when they encountered unparsable input, either lying outside the bounds of the parser's grammar or being actually ungrammatical. However, when dealing with natural language interfaces, or in short with human beings, ill-formedness will occur and has to be dealt with. "While considerable advances have been made in recent years in applied natural language processing, few of the systems that have been constructed have paid sufficient attention to the kinds of deviation that will inevitably occur in their input if they are used in a natural environment" (Hayes and Mouradian 1981, p. 232).

In the early days of CALL, the only sophisticated error treatment available came from NLP practitioners who saw CALL as a possible end application for the error tolerant parsers they were building as part of human-computer interfaces. Thus, the early error diagnosis systems for CALL were often constructed as a secondary output of other NLP applications (such as the system described in (Weischedel and Black 1980)), with the exception of the German Tutor of Weischedel, Voge, and James (1978).

Nevertheless, the approach followed in general NLP systems and in CALL systems towards the treatment of ill-formedness is naturally quite different. For most natural language interfaces, the goal when encountering ill-formedness is to overcome it in order to provide an appropriate reponse and not to block at the ill-formedness site. Therefore, some parsers are underconstrained and are able to give a representation for many ungrammatical sentences as well as grammatical ones. These systems must figure out a way to make some sort of sense of their users' query in order to respond meaningfully. They are not overly concerned with indicating the source of the ill-formedness, unless it is overwhelming and prevents the system from going further. In such case, it might be interesting for the system to indicate the cause of the failure to the users in order for them to rephrase their query.

In the early 1980's, the range of ill-formedness that NLP systems, both in CALL and elsewhere, were able to treat was rather limited. The first acknowledged limitation sprang from the systems themselves, which were often small-scaled and within constrained domains. Weischedel and Black (1980)'s CALL system worked on single texts for each of which "a module of world knowledge specific to the particular text" (Weischedel and Black 1980, p. 98) had to be created. Hayes and Mouradian take pain to emphasize "that FlexP is designed to be used in the interface to a restricted-domain system. As such, it is intended to work from a domain-specific semantic grammar, rather than one suitable for broader classes of input" (Hayes and

Mouradian 1981, p. 236).

The range of ill-formedness coverage attempted by these early systems included co-occurrence violations, ellipsis and extraneous terms, and conjunction for Kwasny and Sondheimer (1981), misspellings, novel words, erroneous segmentation, restarted utterances, elliptical input, interjected phrases, omission and substitutions, agreement failures, idioms and user supplied changes for Hayes and Mouradian (1981). Most of these are also the basics treated by other systems of that period. One can notice that this list includes many *relative* ill-formedness following the definition of Weischedel and Sondheimer (1983, p. 161). Ellipsis, conjunction, novel words, interjected sentences and idioms are not an indication of ungrammaticality. They are, rather, the mark of the proper use of the language within a communication act. These structures are, however, difficult to treat from an NLP standpoint, and one can understand that they would lie outside the normal bounds of the early 1980's systems. For the CALL domain, however, one should restrict correction (and/or diagnosis) to *absolute* ill-formedness, whose treatment at the time covered misspellings, co-occurrence violations (agreement failure, verb selection), erroneous segmentation, omissions and substitutions. None of those seem really directed specifically at language learners, as they are also common errors of native speakers, apart from example (3) from Kwasny and Sondheimer (1981, p. 100), classified by them as a 'co-occurrence violation' and which is akin to the erroneous use of a fixed phrase or idiom.

(3) * I will stay from now under midnight.

It is often assumed that, as Weischedel and Black state it, "in the environment of foreign language instruction, the system is in the unique position of having more vocabulary and syntactic forms that the user and, therefore has more knowledge than the user can express" (Weischedel and Black 1980, p. 98–99), which implies that relative ill-formedness is unlikely to occur. Although this might be true for CALL systems catering to beginner level learners, this is not necessarily the case for advanced level learners, whose knowledge of the language can have multiple sources and differ from a given text book or foreign language method.

However, let us not forget that NLP-based CALL was rather a rarity in the early 1980's. Pusack (1984) cites five different ways of answer processing in CALL:

- Non-evaluation;
- Right/wrong evaluation;
- Pattern markup;

- Error anticipation;
- Parsing.

Pusack devotes most of his efforts to the description of pattern markup, more commonly known under the term 'pattern matching'. This technique was indeed widely used during that period. While requiring a lot of effort to build a complete set of exercises with adequate feedback, this method was rather reliable. It has been used by many systems, such as CLEF (Holmes and Kidd 1980, Paramskas 1993) described in section 2.3.2. Parsing, on the other hand, was often considered not to be advanced enough or requiring too much resources to be used apart from toy software. "Even a simple system capable of doing this [detecting errors of syntax and morphology], however, may use all the resources of a moderate-size computer" (Pusack 1984, p. 63). With the advance of microcomputers, this barrier fell naturally away in the following years.

2.1.2 The late 1980's

In the second half of the 1980's, CALL seems to be emerging as a discipline in its own right. In his introductory address to 'The First Conference on Canadian Computer-Assisted Language Learning', held in 1989, Holmes (1990) reports on the development of CALL, not only in Canada but in the United States and in Great Britain. As a pioneer of CALL, he has seen the emergence of a CALL community which did not exist in the early 1980's. He also talks of the number of CALL research programs which have been funded and launched in the recent past as indicator of the increasing interest in CALL. He notes the variety of CALL activities: "authoring systems, drill-and-practice, simulations, gaming, translation, databases as learning aids, multilingual word processors, writing and composition" (Holmes 1990, p. 5). This diversity, present from the start, is still very much the norm in CALL today. Finally, Holmes mentions the appearance of artificial intelligence techniques including parsing.

In 1992, Hubbard recalls that "in the past few years, CALL has clearly established itself as a separated discipline within language learning: It has its own professional organizations, such as CALICO and the CALL Interest Section of TESOL, journals and newsletters, and a growing number of conferences, monographs and anthologies (including the present one) devoted to it" (Hubbard 1992, p. 40). Indeed, with the creation of CALICO in the mid-1980's, the first EuroCALL constitution in 1988, soon followed by the first issue of the ReCALL journal, and the first Canadian conference on CALL held in 1989, one can sense the increasing interest that CALL, in all its varieties, attracts from the mid-1980's onwards.

Parsing remains an exception during this period. "Even in the area of drill-and-practice there are only a handful of programs which use NLP techniques in order to give students informative feedback, even in a relatively mundane area like the correct use of verb forms" (Bailin 1990, p. 174). Intelligent computer assisted language learning (or instruction, respectively ICALL and ICALI), which includes for some parser-based CALL, is still considered to be very young. "First of all, ICALI is in no way a large field: there are, in fact, relatively few projects devoted to the development of ICALI software. Second, it has not created a substantial body of theory: the work is still in its infancy. Finally, it has not yet produced much significant software: what software there is often hints at far more than it can deliver" (Bailin and Levin 1989, p. 3).

This might be explained by two factors. First of all, the second half of the 1980's saw the emergence of CALL software taking into account communicative approaches to language learning. Guberman talks about "open-minded software" (Guberman 1990, p. 33) where the focus is not on grammar and form but on meaning and content. She describes several such products for English, French, and Italian. These are 'enjoyable' software which do not seem to be teaching and thus might be better accepted by language learners, and which, moreover, do not necessarily require a diagnosis or a correction of the users' input. The growing interest in communicative approaches to language learning and its arrival in the CALL domain implied a shift of focus away from grammar and the treatment of ill-formed input by the software. Learners must not be hindered in their practice of the language by warnings about the ungrammaticality of their productions.

Secondly, this period saw the emergence of the personal microcomputer. The microcomputer brought affordable machines to language teachers, who might not have had access to the mainframes guarded by computer scientists. At the same time, language learners became more used to computers and thus less afraid to work with them. CALL software users were not necessarily learning to use a computer at the same time they were trying to use the software and learn the language. The increasing availability of microcomputers is having an impact on CALL designers and authors. Language teachers have now the means to devise their own CALL material, as it is well attested in Craven, Sinyor, and Paramskas (1990). However, most language teachers lack the computer science competencies to create NLP tools for error diagnosis.

2.1.3 The early 1990's

CALL continues to affirm its position as a scientific domain in the early 1990's with the EuroCALL annual conferences, the first one being held in 1993. EuroCALL has always served the whole of the CALL discipline and many areas are represented, including NLP in CALL.

The early 1990's also saw several important publications in the domain of CALL, including NLP-based CALL, under numerous different acronyms.

Bailin edited a special issue of the CALICO Journal on ICALI in the Fall of 1991 (Bailin 1991). Swartz and Yazdani (1992) published a collection of papers in book format in 1992, and Holland, Kapland, and Sams (1995) another a few years later. This latest collection of CALL papers covers a wide range of CALL domains, in which NLP-based CALL is particularly well represented.

NLP-based CALL systems were still far from the norm in the early 1990's. Some CALL practitioners advocated it. "A sophisticated foreign language training system must be able to evaluate free text inputs from the student, markedly distinguishing it from systems that can only handle true-false, multiple choice, or canned fill-ins. Sole reliance on simple exercises will not support acquisition and maintenance of real communication in the foreign language" (Criswell, Byrnes, and Pfister 1992, p. 312). Others did not believe in the rapid use of NLP tools in CALL. "In education, sophisticated interactive grammar-checkers built on instructional parsing systems might eventually be used for direct students instruction, but we do not expect such use will soon be widespread" (Loritz 1992, p. 17).

During that period, Holland, Maisano, Alderks, and Martin (1993) also deem fit to write an apology of NLP for CALL. NLP is seen as "relatively new to CALL" (Holland et al. 1993, p. 28) and the authors' aim is to tell their readers what NLP is, how parsers can be used in CALL, what limitations there are to this technology, and how to overcome them. Still quite unknown to many CALL practitioners of the early 1990's, parsers in CALL already exist, however, and Holland et al. (1993) are able to cite no less than five such systems or family of systems.

Progress of NLP-based CALL in the CALL community is rather slow, due in part to a lack of maturity of the tools then available. Nevertheless, the advance of NLP-based CALL is also partly a problem of publicity. The teacher population who would supposedly introduce their students to NLP-based CALL is not necessarily aware of the existence of such tools.

2.1.4 The late 1990's and beyond

The field continued to expand in the late 1990's with the appearance of two web-based scientific journals for the CALL community in 1997 (ALSIC[2] and Language Learning Technology[3]) and the first WorldCALL conference in Australia in 1998.

The shift to Internet does not appear only with web-based journals. The topic of many papers and conference presentations is on the use of Internet, in all its forms (email, world wide web, chats, ...) to improve language learning. "Web-based CALL systems are increasing day by day" (Yang and

[2]http://www.alsic.org/ (last accessed on April 24th, 2003)
[3]http://llt.msu.edu/ (last accessed on April 24th, 2003)

Akahori 1999, p. 61). A cursory look at the presentation summaries of the 1999 EuroCALL conference held in Besançon, France (Chanier 1999), is enough to be convinced.

Despite the growing importance of the CALL community, NLP is still not widely used. "Although CALL employs the computer to assist in language teaching and in language self-study, most CALL programs make little essential use of language technology, exploiting instead hypertext, digital audio and video, (simple) database technology and network communications." (Nerbonne, Dokter, and Smit 1998, p. 544). "The fact remains that it is still not possible to have a language learner type even a short composition into a computer and to have the computer correct anything but the most basic linguistic errors such as verb and adjective agreements and misspellings" (Richmond 1999, p. 303). Thus, error detection, diagnosis and correction, if any can be found, are most often still employing simple techniques such as pattern matching.

In summary, although NLP-based CALL exists, it is still not widely used in the CALL community at the time of writing. Whatever systems one finds using this technology are still mostly research prototypes not commercialized and thus not available to most foreign language teachers and learners.

2.2 Spell checkers

Although spell checkers are not really part of the topic of this dissertation, we briefly describe them here for two reasons. Firstly, it is a domain where language learners are also prone to make errors and which influences parsing. Secondly, they were the first NLP tools to be widely available and used by the general public in word processors. They treat words in isolation, but they are satisfying to their users because they do what they are supposed to, detect misspelled words, and they do it quite fast.

Peterson (1980) cites a number of different techniques for spell checking, which were either currently used or seen as possible research leads. They include ordering the tokens of a text by frequency, detecting rare digrams and trigrams as potential misspellings, dictionary lookup, grapheme to phoneme to grapheme transcriptions, and affix analysis. Many of these techniques are currently still used in spell checkers, that is more than 20 years later.

The importance to treat phonographic errors is stressed in Véronis (1988), although the author writes about native speaker data. He proposes that "a whole *grapheme*, which can be more than one letter long, can be replaced by another grapheme having the same phonetic value" (Véronis 1988, p. 709, emphasis in original).

The spell checker designed by Vosse (1992) uses trigram and triphone analysis, with a ranking and scoring mechanism. Moreover, Vosse uses syntactic filtering to eliminate some of the alternative spelling corrections. Once a set of possible corrections has been found for an unknown word, the sentence is parsed with all the correction alternatives. Only the alternatives resulting in a correct parse, recuperated by inspecting the parse trees, are made available to the users.

Courtin, Dujardin, Kowarski, Genthial, and de Lima (1991) propose a detection/correction system for lexical and morpho-syntactic errors. The lexical level, the closest to a spell checker, uses three detection techniques: skeleton key, phonetics and morphological parsing. Skeleton keys are described in Pollock and Zamora (1984). The main idea of skeleton keys is to "generate a similarity key for each word in the dictionary and then sort the dictionary in key order. A misspelling is then corrected by locating words whose keys collate most closely to the key of the misspelling and selecting the plausible correction(s) from these" (Pollock and Zamora 1984, p. 359). The exact form of the key varies from one implementation to the other. "Modern correctors generally use the reference word to calculate a key which is then used to search the lexicon" (Letellier and Fournier 1990, p. 394). The second technique used by Courtin et al. (1991) computes, for any given misspelled string, "all the possible phonetic transcriptions associated with this string. (...) From these phonetic transcriptions, we generate written forms which give us proposals of corrections for the misspelled word" (Courtin et al. 1991, p. 159–160). Finally, morphological parsing is used to correct wrong inflectional endings, such as the use of the regular plural morpheme on an irregular noun. These techniques are usually activated in the order of presentation, but the authors indicate that other orders might be more interesting depending on the user population. In particular, "for a text written by a child or a foreigner, it may seem preferable to start with phonetic correction" (Courtin et al. 1991, p. 169).

Spelling correction with error-tolerant finite-state recognizers is proposed by Oflazer (1996). In an agglutinative language like Turkish, it is impossible to base a spell checker on a word list, as it is often done for languages such as English or French, given the highly combinatorial nature of possible suffixes. Roots and affixes are encoded in a finite-state recognizer. "One needs to find *all* paths from the start node to one of the final nodes, so that when the labels on the links along a path are concatenated, the resulting string is within a given edit distance threshold t, of the (erroneous) input string" (Oflazer 1996, p. 75, emphasis in original). For efficiency reasons, a path is pruned during the search as soon as it is over the threshold. "Insertions, deletions, replacements, and transpositions" (Oflazer 1996, p. 84) can be treated this way, which, once again, might not be sufficient for language learners.

"Relevons qu'il n'existe pas, à notre connaissance, de correcteurs orthographiques commerciaux munis d'algorithmes d'identification et de tri des solutions conçus spécialement pour les utilisateur de langue seconde" (Cornu 1997, p. 29). Some spell checkers are said to cater *also* to language learners (Courtin et al. 1991), but it is not their main public. Thus, language learners have to do with tools designed for native speakers and not adapted to their specific needs. Commercial tools are nevertheless usable by language learners as they point out any word which is not part of their lexicon. What they are less well equipped to do is provide correction proposal tailored to the spelling errors of language learners.

After this brief introduction to spell checkers, we now direct our attention in the next sections to several diagnosis techniques used for grammar checking, starting with pattern matching.

2.3 Pattern matching

Pattern matching was not invented for error diagnosis, but it has been widely used in CALL as it is a relatively easy way to provide some form of feedback to the users. We give a description of pattern matching and then describe several CALL systems using pattern matching to diagnose errors.

2.3.1 Description

In CALL, the author/teacher provides the system with a pattern, for example the correct answer to a question, which must then be matched to a search space, the learner's input. In the simplest form of pattern matching, the pattern to be found and the search space must be exactly identical. "The pattern-matching problem can be characterized as a searching problem with the pattern as the key" (Sedgewick 1988, p. 278). The matching algorithm must then verify that the current element of the pattern (a character, a word) matches that of the search space. One repeats this step until the end of the pattern, the end of the search space or the mismatch of an element. The match fails in case of a mismatched element, or of the end of the pattern and the search space not being reached at the same time; otherwise, it succeeds.

(4)a. Pattern: My tailor is rich.

b. Search space 1: My tailor is rich.

c. Search space 2: My taylor is rich.

Pattern (4a) can be found in search space (4b) as they are identical, but not in (4c) because of the spelling error on 'taylor'.

To complicate things somewhat, the pattern can be a subpart of the search space, in which case the pattern could be matched more than once within the same search space. The algorithm must be refined to start a new search for every element of the search space. "The algorithm looks for the leftmost substring in the text string which matches the pattern description by scanning the text string from left to right, testing at each position whether there is a substring beginning at that position which matches the pattern description"(Sedgewick 1988, p. 295).

Feedback is performed by a message, or message reference, attached to each pattern entered by the author/teacher of a given question. With pattern matching as described so far, if several correct answers are possible, all must be entered in full. Likewise, if incorrect answers are provided, they must all be entered in full with their appropriate associated feedbacks. This is naturally quite time consuming, and typographical or other mistakes due to a lack of attention are likely to crop up. Therefore, in order to reduce the work of entering multiple full patterns, languages have been defined to describe complex patterns. These languages allow for concatenation, alternative and repetition. "Various additions are commonly made in actual systems for convenience. For example, –A might mean 'match any character *except* A'. This *not* operation is the same as an *or* involving all the characters except A but is much easier to use. Similarly, '?' might mean 'match any letter'" (Sedgewick 1988, p. 294, emphasis in original). Using such languages allows authors/teachers to enter their correct and incorrect answers in less time and with less risk. The pattern matching algorithm must be able to take alternatives into consideration and non-deterministic finite state automata are often used for this purpose. This pattern language remains quite simple and can be used by any one without prior computer knowledge.

There are several ways to use pattern matching in CALL. One can either look for correct complete sentences, or parts of sentences, or look for erroneous sentences or erroneous parts of sentences. This choice has interesting implications for diagnosis and feedback. On the one hand, if the option to check for correctness is taken, the system will not be able to provide appropriate feedback on specific errors to the learners. On the other hand, if one looks for specific errors, there is the risk to miss flagging some unexpected incorrect learner production and to accept it because it does not coincide with any error pattern. Using patterns to detect both correct sentences and incorrect parts of sentences seems necessary. The patterns detecting errors can then be launched only if the pattern for the correct answer was not found.

Pattern matching provides relatively good results in very constrained domains and simple tasks. Even with these restrictions, it is very labor intensive as both correct and incorrect answers must be provided. Moreover, "pattern matching is not entirely adequate for natural-language processing because natural-language syntactic generalizations depend on the grouping of words into constituents, not just on their superficial linear order" (Levin, Evans, and Gates 1991, p. 52).

2.3.2 Systems using pattern matching

The information about systems described in this section was gathered through articles in scientific papers. Only one system, CLEF, was available for testing, which explains that it is the first system discussed and has the most extensive description provided. CLEF's description is followed by information on three other systems ordered chronologically by date of publication. This section ends by mentioning others systems using pattern matching for which too little information was found to merit a separate entry.

CLEF

One system using pattern matching techniques is CLEF (Computer assisted Learning Exercises for French). It was first developed in Canada in 1978 and released by the CLEF group in 1986. Since 1996, it is distributed by Camsoft in the United Kingdom. CLEF runs under MS-Dos and therefore looks quite old-fashioned to people used to the multimedia era of the beginning of the 21st century.

"CLEF offre une révision de la grammaire française depuis le niveau débutant jusqu'au niveau intermédiaire: y sont compris dans ses 62 leçons, 62 explications grammaticales et près de 250 exercices de types variés; la rétroaction (commentaires pour guider l'étudiant vers la bonne réponse) est encore aujourd'hui une des plus sophistiquées" (Paramskas 1993, p. 67). "Each lesson begins with a brief optional review of the grammar point. Then, in an effort to minimize vocabulary problems, the major lexical materials to be encountered in the exercises are presented" (Holmes and Kidd 1980, p. 9). The exercises can be done in any order and redone as many times as one wants. For most exercises, the software gives a second chance to the users if they were wrong the first time round, providing some feedback. If the second try is also incorrect, the solution is given.

CLEF was tested as a black box to try to see if pattern matching was actually used in the system as was expected. Not all the lessons and exercises were accessed, but hopefully enough of them to cover all the exercise types. Exercise types include multiple choice questions; fill-in-the-blanks with or without a list of words supplied, with or without having to transform them; reordering of phrases to build a sentence; sentence transformation (usually, part of the transformed sentence is already supplied which limits the amount of typing the user needs to do); and finding the correct position in the sentence for a specific element. The user has to type in either single letters to indicate a choice, numbers to show positions, or a few words. It is doubtful that the user ever has to type more than two or three words at a time.

Testing was conducted by a French native speaker making errors on purpose to both see the kind of feedback provided and try to overpass the limits of the system, which proved relatively difficult given the highly restricted environment. In no place is it absolutely necessary to use any technique beyond pattern matching in order to provide the feedback displayed. Some instances of feedback reinforce our opinion that pattern matching is indeed the technique used, as illustrated below.

In the following examples, text displayed by CLEF will be in `this font`, *italic* is used for what the user types in, **bold** for the expected answer and an arrow ($\Longrightarrow$) precedes feedback.

Lesson 10, exercise (a), asks the user to type in the provided adjective with the correct agreement ending in order to complete the sentence. There is only one correct solution and few possible incorrect ones if the user follows the guidelines. Using another adjective than the one provided has the following effect.

(5) `(joli) La chemise est` *rose*.
(beautiful) The shirt is pink.
jolie
$\Longrightarrow$ Ce n'est pas correct.
It is not correct

This (rather) uninformative feedback was only to be expected. Indeed, the answer is not correct as another adjective was used. The system does not try to check if the sentence is still grammatical. In exercise (d) of the same lesson where one must find antonym adjectives, one can see that punctuation is not properly handled.

(6) `Les enfants sont petits, mais les adultes sont` *grands!*.
The children are small, but the adults are tall.
grands
$\Longrightarrow$ Le radical **grand** est correct mais après il y a une faute d'orthographe.
The radical tall is correct, but there is a spelling mistake afterwards.

The software finds `grand` (tall) but does not know how to treat the `s!` sequence. It thus announces an error, even though the user's answer was adequate, because extra punctuation was added. Even more interesting are perhaps the next two examples, taken from exercise (d) of Lesson 9. The instructions require the user to choose the appropriate verb from a list and to inflect it according to the given sentence. The user, however, has typed in a word which does not belong to the list.

(7) `Charlie Brown et Lucy` *proposent* `de la limonade devant la maison.`
Charlie Brown and Lucy offer of the lemonade in-front-of the house.
vendent
$\Longrightarrow$ La forme est correcte mais le sens est incorrect.
The form is correct but the meaning is not-correct.

(8) `Je` ***`rapporte`*** `ce livre à la biliothèque, je prends un autre livre.`
I bring-back this book to the library, I take an other book.
rends
$\Longrightarrow$ Le sens et la forme sont incorrects.
The meaning and the form are not-correct.

In example (7), one can hope that the user's input has been recognized to be in 3rd person plural and therefore in the correct form, although an incorrect verb was typed in. Example (8) wipes out this illusion. Indeed, the sentence seems perfectly acceptable for both form and content. CLEF does not recognize that `rapporte` (bring-back) is a 1st person singular. `rapporter` (to bring-back) was not part of the list of supplied verbs. The exercise is on verbs ending in `-re` (prendre, vendre, etc.) and their conjugation. CLEF is obviously expecting the verb in (8) to end in `-ds`. Not finding this form, it considers it as not correct. Moreover, as the verb is not the expected one, the meaning is also considered erroneous. A true analysis of the sentence would have permitted to detect that (at least) the form was correct. Detecting that the meaning was also correct would have required either an extended list of possible verbs or a strong semantic component.

Let us not, however, downplay the possibilities of pattern matching. If the exercise instructions are followed and a verb is selected from the provided list, the feedback is coherent. CLEF uses pattern matching intelligently and to a high degree of refinement. Exercise (a) of lesson 31 is a text transformation exercise on the *passé composé* of verbs conjugated with *être*. The whole text is at the first person singular. Before correcting the first transformation from the present to the *passé composé*, the software reminds the user that there must be agreement between the subject and the past participle and asks whether the user is a man or a woman. Depending on this answer, all the past participles in this exercise must be in the masculine or the feminine form.

CLEF does what it is meant to do rather well. Regardless of the variety, all the exercise types are set in a very restrained domain which does not allow the users the degree of freedom and creativity that would enable them to actually produce full sentences in the target language.

Liou's English Grammar Checker

Liou's English Grammar Checker (Liou 1991) is described as using error patterns to detect errors and an augmented transition network (ATN) parser. This made us wonder at first whether we were dealing with an actual pattern matching diagnosis technique or if the system was using an ATN to detect errors. However, "the program activates the error pattern matching process before the parser" (Liou 1991, p. 65) and the error patterns include actual words such as 'although' and 'but'. These combine to reinforce the interpretation that an actual pattern matching procedure is invoked, although there are not enough details in the paper to be actually certain of the fact. The parser is activated only if no error pattern is found, simply in order to confirm that the sentence under scrutiny is actually part of the set of lawful and acceptable sentences as described by the grammar.

Parsing a sentence if no error pattern is found is an interesting, though costly, option to prevent unexpected incorrect productions to be accepted.

CALLE

Rypa and Feurman (1995) describe a noteworthy use of pattern matching. Their system contains detectors to look for particular linguistic constructions, which are not necessarily errors. The most interesting part of these detectors is their detector pattern. "The pattern is a template to be matched against the parse solutions. (...) The pattern consists of an equation, a constraint, or a Boolean combination of equations and constraints involving constituent variables" (Rypa and Feurman 1995, p. 70). The detector patterns are matched to the result of the parse of a sentence. The parse itself involves a full lexical-functional grammar. This pattern matching variation is particular as far as the patterns to be matched are not raw text, but the results of a complex NLP activity. If a pattern finds a match, then the message part of the detector is presented to the user.

In the project presented, the detectors were used to provide explanations on the sentence under scrutiny. This sentence was supposedly grammatical. At the end of their article, Rypa and Feurman remind their readers that one of the goals of the original CALLE project was to develop an error diagnosis system (Feuerman, Marshall, Newman, and Rypa 1987). "At that time, extensive LISP programming was required to determine the error present in a parse of a student-composed sentence. The new feature detectors from the CALLE90 project can be utilized to detect errors in the students' language, thereby eliminating the need for programming LISP code for this purpose" (Rypa and Feurman 1995, p. 76). Whether this was actually done or not is unfortunately not part of their article.

VINCI

Pattern matching did not disappear with the advancement of parsing. It is still used nowadays, as it is proven by the description of the VINCI system in the year 2000 (Levison, Lessard, and Walker 2000). VINCI uses a version of pattern matching to compare a system generated correct response to the actual response provided by the learner. The algorithm uses approximate string matching and a two-level approach. "In our error-detection process, the upper level uses an approximate string matching algorithm to match the string of words in the correct response to the string of words in the student reply. (...) The lower level of our process arises when the approximate string matching algorithm compares a word of the correct string with a word of the actual response" (Levison et al. 2000, p. 316-317).

While the system is able to treat in this way a range of errors, including word insertion or deletion errors, word order errors, typing errors, phonological errors, synonyms, and morphological (both derivational and inflectional) errors, on the syntactic front the authors are forced to admit that "if several alternative syntactic forms are reasonable, these can be included as separate correct responses" (Levison et al. 2000, p. 317). It is unclear whether these additional correct responses are also generated by the system and automatically included, or whether this work has to be done by hand, which would tremendously lower the interest of the VINCI system.

Although diagnoses are given in the article for five possible errors on the same sentence, no result on actual learner data is provided. There is therefore no real evaluation of the system, as far as one can tell. Moreover, the examples given tend to point out to very restrained contexts which do not give the learners the opportunity to be creative in their productions, as was to be expected from a pattern matching based system.

Other mentions

The Elementary Language Study Experiment (ELSE) was first designed by Allen in 1970. It was abandoned for a number of years because of portability difficulties between different mainframes computers, and reprogrammed in 1990 on IBM-compatbile microcomputers under a slightly different name but keeping the same acronym: Elementary Language Study Exerciser (Allen 1997). Allen does not explain the answer checking mechanism that ELSE is using, but, from the examples provided, it seems to be a pattern matching technique.

Hull, Ball, Fox, Levin, and McCutchen (1987) propose a grammar checker for college students native speakers enrolled in a remedial writing course based on 'augmented pattern-matching' "a program which will search, not only for literal strings of characters, such as individual words, but also for part-of-speech patterns" (Hull et al. 1987, p. 108). Even with their augmented version of pattern matching, they recognize that "there are several clear limitations to pattern-matching as an approach to detecting errors in natural language texts" (Hull et al. 1987, p. 115–116) and one of their further research plans includes writing a natural language parser.

The DRIP system, briefly described in Güvenir (1992), although using NLP techniques for sentence generation, is employing a very simple pattern matching mechanism to detect errors. "The learner's sentence is compared to the sentence obtained at the end of each pass. Only the terminal symbols, that is letters, are matched" (Güvenir 1992, p. 290). It is surprising to see that matching the learner's translation to the computer generated sentence is done character by character and that no specific linguistic knowledge is required for the diagnosis process, while it is necessarily available in order for the generation part of the system to work appropriately.

2.4 Finite State Automata

Many parsers are implemented using finite state automata (FSA) (such as Weischedel and Black (1980)'s and Kwasny and Sondheimer (1981)'s). Our purpose in this section is not to describe those parsers, but rather to describe the error detection technique which makes use of FSA to locate errors within texts, without necessarily parsing the whole input.

2.4.1 Description

To our knowledge, only one system uses this formalism, which is the ARCTA prototype, of the *Laboratoire de traitement du langage et de la parole*, Université of Neuchâtel, Switzerland. We can also note the lack of references to other systems using automata to detect errors without parsing the sentence in Kübler and Cornu (1994) and Cornu (1997). This reinforces our idea that this technique is not widely used.

Lacking a diversity of view points and of variants of the technique, it is difficult to give a description of the technique as such by basing oneself on only one system. Other systems, if they exist, might be using automata in a completely different way. Therefore, we are not in a position to give a general description of the diagnosis technique by automata in this section. A specific description of the technique used in the ARCTA project is nevertheless provided.

2.4.2 The ARCTA prototype

The ARCTA prototype grammar checker (see (Tschumi et al. 1994; Tschichold et al. 1994; Cornu et al. 1996) for a complete description of the ARCTA project) uses finite state automata to detect errors made by French-speakers writing in English. ARCTA's automata are a modified version of FSA and of Winograd's augmented transition networks (ATN) (Winograd 1983). "Un automate peut être décrit comme une série de conditions et d'actions qui s'appliquent à des suites d'éléments du texte, comme des mots, des signes de ponctuation ou des syntagmes nominaux" (Cornu 1997, p. 207). First of all, these modified automata have an anchor arc. This arc is the first one to be verified in the automata. If it succeeds, then the arcs to the right are checked, followed by the arcs to the left. This anchoring allows the authors to link automata to specific error types and to traverse the arcs in both directions. There is no backtracking and no recursion. Other adaptations of FSA are the access to lists of words grouped by common features (e.g. temporal nouns: afternoon, Christmas); the use of registers to store persistent information in case of success of the automata; the possibility to create new features for words and phrases; and the possibility to make reference to nominal phrases.

Moreover, the "automata operate on three levels. The first level is a preprocessing one and is concerned only with extracting new data from existing structures (Data Extraction Automata, DEA). The second level is optional and contains filter automata (FA). These identify sentence patterns in order to determine when to activate the third-level which contains detection automata (DA)" (Kübler and Cornu 1994, p. 240). They propose the example of a filter automaton checking for a verb such as 'give' which allows a shift of the direct and indirect object, in which case the indirect object loses its preposition. If such a verb is found by the filter automaton, detection automata are launched to check whether there was a shift and, if so, whether the preposition is present. If it is, an error is flagged, as in (9) (Kübler and Cornu 1994, p. 243).

(9) * He gives to his friend a wonderful present.
Error message.

For every level of automata, there is a list of automata to be launched in a specific order. "Cet ordre est d'une grande importance, car il permet un classement rigoureux des automates: le plus complexe est exécuté en premier, le plus simple en dernier" (Cornu 1997, p. 212). Thus, a simple structure will not be recognized if it is part of a more complex one. To prevent simpler automata to fire, the elements used by any successful automata of the same level are removed from the list of accessible elements to the other automata. "Lorsqu'un automate 'passe', il empêche tous les autres automates de la liste d'utiliser les mots qui correspondent à ses arcs" (Cornu 1997, p. 213). However, and as nothing is said to the contrary, we assume that it is the authors of the automata which must classify them by order of complexity.

Extensive results, together with a discussion of these results, are provided in Cornu (1997, p. 268 ff.). Diagnosed error types are: subject-verb agreement; passive construction; choice of prepositions in temporal complements; use of the continuous tenses; subject-verb order in indirect discourse; verbal form in infinitival clauses; noun agreement; verb tenses with temporal complements; and relative position of verbs and adverbs. The prototype, tested on a corpus and on test sentences, is able to correct roughly 80% of the errors, which seems a pretty good score. Overflagging is very rare: 22 occurrences for over 1'500 sentences. A comparison with commercialized checkers confirms the good results of the ARCTA prototype (Cornu et al. 1996).

Kübler and Cornu recognize that their "approach is less efficient in detecting errors linked with complex grammatical structures" (Kübler and Cornu 1994, p. 245). However, they emphasize that their approach with three levels of automata reduces the risk of false positives (or overflagging). "Limiting overflagging is particularly important in the context of L2 texts as stopping on a non-error may mislead the user whose mother-tongue is not English" (Tschumi et al. 1994, p. 224).

Moreover, Cornu indicates that the technique does not provide a full syntactic analysis of the text under scrutiny. This is seen as an advantage because "nous pensons pouvoir éviter les cas où des données incomplètes lors de l'analyse provoquent des malfonctionnements" (Cornu 1997, p. 199). Although this point is very important, the lack of a full analysis also has some disadvantages in the CALL context, as one would like to be able to offer the users information on the syntactic structure of the sentence even if it is ungrammatical. Having to parse the sentence once the error detection phase is over would be a waste of time.

2.5 Error grammars

This section is divided into two parts. The first one describes what error grammars are and the second details some systems using an error grammar.

2.5.1 Description

Use of an error grammar implies a rule-based system which has a set of grammar rules to treat grammatical input and a parsing algorithm separate from the grammar. "Parsing ill-formed input is problematic for a rule-based system. (...) A standard solution to this problem is to introduce more rules designed specifically to handle ill-formedness. This is a familiar approach in ICALL" (Matthews 1993, p. 19). Sanders and Sanders (1989) define an error grammar as "a grammar of sentence structures representing typical errors. When the input fails the correct grammar, it will be passed to the error grammar. If it matches one of the structures in this grammar, the error message associated with that structure can be passed to the student" (Sanders and Sanders 1989, p. 18). Thus, the rules of the correct grammar must be activated before the rules contained in the error grammar, also called error rules or mal-rules.

Only errors planned in advance can be detected with error grammars, as a new (set of) error rule(s) must be included in the grammar for each error type. "It is also a problematic exercise in writing enough rules to handle all possible ill-formed input. Finally, there is also the problem in that simply producing a rule to allow through an ill-formed example does not provide an explanation for that failure" (Matthews 1993, p. 19). Indeed, error messages must be attached to the error rules to be displayed (or stored for later display) whenever an error rule succeeds.

Conditions for an error rule to apply can be more or less specific, defining the range of constructions it applies to. "A high degree of granularity can only be achieved by means of very special rules. This could result in overly specific error rules which only cover very few examples. A central task in designing error rules is consequently to find a level which is general enough to have several instances covered by the same error rule, and is specific enough to be able to give adequate feedback to the learner" (Murphy, Krüger, and Grieszl 1998, p. 65). Granularity is, of course, important: the more specific one can be, the more adequate and relevant the feedback to the users. The trade-off is in the time needed to conceive the multitude of very specific error rules. Granularity depends partly on the level of representation used in the rules. Including many specific words in an error rule rather than parts of speech (POS) renders a rule more granular and specific but demands more rules to treat errors where other words are used.

Thus, "an error grammar is potentially at least as large as the main grammar. The more complete its coverage, the more likely it is to adversely affect efficiency, i.e. parse times. Program designers would be well advised to choose carefully which errors to pinpoint in order to maximize the relationship between completeness and efficiency" (Sanders 1991, p. 77). Indeed, users rarely agree to wait for very long before they receive feedback from a system. It might be better to have a less performing but more efficient system, actually used by learners, that a very sophisticated but slow system, which nobody uses because its reaction time is too long.

2.5.2 Systems using error grammars

Five systems using error grammars are described in this section. They are ordered by the date of the earliest cited publication. The section ends by mentioning several other systems using error grammars, but of which not much is known.

German Tutor

Weischedel, Voge, and James (1978) were, to our knowledge, the first to propose a CALL system relying on NLP with their German Tutor. Common word order mistakes are anticipated by the adjunction of error rules, distinct from the grammatical ones. "The syntactic analyzer is designed to find its way through mistake-ridden student prose: in fact, it has rules for common student mistakes, and when it can produce a correct parse only by invoking one of its mistake-rules, it makes a note that a particular mistake has been detected" (Mulford 1989, p. 35–36). These pioneers' prototype had only a small grammar and a vocabulary of less than 200 words, but it demonstrated the possibility of using an error grammar to detect some syntactic error types. Some agreement and semantic errors were also treated.

Schwind's error analysis and explanation system

Schwind's error analysis and explanation system (EAES) (Schwind 1986, Schwind 1988, Schwind 1995, see also section 2.6.2) uses error rules to detect three types of errors: low-level and high-level syntactic errors, and semantic errors. "Low level syntactic errors involve the omission or the addition of functional words such as articles or prepositions. High level syntactic errors involve the permutation of groups of words" (Schwind 1995, p. 315). Error rules are added to the rule set for missing and additional prepositions and articles as well as for several types of ungrammatical word order, such as the misplacement of the verbs or of adjectives. As for semantic errors, "the only type of semantic errors on which we have been working so far concerns the violation of semantic restrictions on verbs and their complements" (Schwind 1988, p. 612). A specific rule for semantic mismatch is also part of the grammar rules.

The FROG - FGA - LINGER - ISCA family

The FROG (French RObust Grammar checker) (Imlah and du Boulay 1985), FGA (French Grammar Analyser) (Barchan, Woodmansee, and Yazdani 1986), LINGER (Language INdependent Grammatical Error Reporter (Yazdani and Uren 1988, Yazdani 1991), and ISCA (Interactive Sentence Constructor and Analyser) (Bolt and Yazdani 1998) systems belong to the same family of grammar checkers. Each new system is built, at least in part, on its predecessor. "[FGA] began as an attempt at a rational reconstruction of Imlah and du Boulay's FROG" (Barchan et al. 1986, p. 30). "Attempts at making FGA more general, and easier to use by people without programming experience, led to a new system called LINGER" (Yazdani 1991, p. 109–110). "The development of ISCA resulted from concerns about the existing conception, architecture and performance of LINGER and the difficulty of adapting or refining it to achieve better performance" (Bolt and Yazdani 1998, p. 69).

In both FROG and FGA "expected incorrect structures can be anticipated and built into the grammar with an appropriate error message tag" (Barchan et al. 1986, p. 32). This classifies them into the group of grammar checkers using error grammars. Once the syntactic structure is built, errors, stored on the tree, are reported. "The error reporter performs a traversal of the sentence structure(s) and makes a check for each node, reporting any errors found, possibly with a correction" (Imlah and du Boulay 1985, p. 145).

In LINGER, "the shell enjoys the benefit of some language independent heuristics, i.e. very basic mistakes that are likely to affect any language. Current heuristics include: word is in wrong place; word is missing; word should not be present; one word should not be present and another word is missing" (Yazdani 1991, p. 110). These heuristics take care of many structural errors, including the wrong positioning of adjectives and missing prepositions, and thus will allow a syntactic tree to be built although all the grammatical rules might not be respected. The syntactic tree then undergoes a series of checks to verify morpho-syntactic properties "such as appropriate numbers, gender and so on" (Yazdani 1991, p. 111). LINGER is therefore shifting form the error grammar paradigm into the delayed feature checking paradigm (see section 2.7.1).

Information on the error detection process in ISCA (Bolt and Yazdani 1998) is rather scarce. It seems, though, that like with LINGER, a syntactic structure is built first and that error reporting is done on the basis of checks on that structure.

Quantitative results are not given for this family of checkers, although each one is said to be better than its predecessor. ISCA is compared to no less than fourteen other systems on a few sample sentences and is performing better than the competition on them (Bolt and Yazdani 1998). However, these sentences are almost certainly not authentic learner productions and might have been selected and/or created by the ISCA authors, thus possibly biasing the results.

ALICE and ALICE-chan

ALICE is a generic environment for building CALL systems. Two prototype applications have been implemented: one for Spanish and one for Japanese. Parsing in the Spanish prototype uses two sets of rules, "including rules for parsing erroneous sentences" (Levin, Evans, and Gates 1991, p. 41), thus clearly identifying ALICE within the group of systems using error grammars.

ALICE-chan (Levin and Evans 1995) is the Japanese version of ALICE for beginner-level. Its NLP component is composed of several modules, including a segmenter and morphological analyzer, a parser, a mapper and a matcher. Errors in users' productions can be diagnosed in each of these modules through different techniques. We will concentrate in this section on the errors diagnosed during the parsing phase, which uses specific rules to handle ill-formed input. "During parsing, special grammar rules are designed to parse error-full structures and raise error flags" (Levin and Evans 1995, p. 86). Although the term of 'error-grammar' or 'error-rule' is not mentioned, it is quite clear that this technique is indeed applied here. Errors to be detected through this means are: wrong past tense of adjective (acting as predicate), wrong use of adjective, wrong structure for 'na nominal' modifying noun, wrong negation of noun, and disagreement of noun and classifier (from figure 5.9 (Levin and Evans 1995, p. 88)). The set of errors diagnosed seem rather small but, without knowing Japanese, it is impossible to determine whether they are actually crucial errors often produced by beginner learners of Japanese. Moreover, other types of errors being treated by other NLP modules, this set might be quite adequate for the intended use. No experiment results on actual learner data was provided.

Two caveats are given about ALICE. "When the ALICE parser encounters an error that was not predicted and included in the grammar, the parse will fail. (...) [The system] will only detect errors in sentences that it can parse completely" (Levin et al. 1991, p. 42). This shows the limitations of error grammars, as seen by its own practitioners. Indeed, in their extension plans for the NLP module, Levin et al. (1991, p. 51) mention other diagnosis techniques such as pattern matching (cautiously pointing out its additional problemas) and constraint relaxation.

ICICLE

The Interactive Computer Identification and Correction of Language Errors (ICICLE) is a CALL system focusing on the needs of American Sign Language (ASL) native signers learning English. "The system recognizes errors by using mal-rules (also called 'error production rules') which extend the language accepted by the grammar to include sentences containing the specified errors" (Schneider and McCoy 1998, p. 1198). The major problem faced is the omission of constituents as a result of negative transfer from ASL. Determiners and the copula are often missing from the sentences produced by this group of learners. Handling omissions with error rules is difficult because the combination of several omission rules in the parse of a single sentence could (almost) allow the system to recognize a blank space as a grammatical sentence. The solution implemented in ICICLE involves specific procedures which check constituents for missing elements and deletes those with too many or incompatible missing elements.

Although grammar coverage is not very extensive (coordination and clause adjunction are not treated, for example) results are interesting. "If we eliminate the nine sentences that are actually grammatical in isolation, we are left with 70 sentences, of which 44 (63%) have parses with the expected error, three (4%) are wrongly accepted as grammatical, and 23 (33%) do not parse" (Schneider and McCoy 1998, p. 1203). Moreover, the system wrongly detected errors in only 3% of grammatical sentences. It seems that, by improving the coverage of the grammar, more sentences will parse, thus improving the detection rate which is rather low.

Other mentions

Articles about several systems mention 'in passing' that they use an error grammar as part of their diagnosis module. Felshin explains that "structural errors such as misorderings are anticipated by addition of extra rules" (Felshin 1995, p. 257) in the Athena project. EPISTLE (see Miller, Heidorn, and Jensen (1981), among others, and section 2.6.2 for a more detailed description) also employs some error rules as part of its overall diagnosis methodology.

Syncheck (Sanders and Sanders 1987, Sanders 1991) uses an error grammar to detect morphological and syntactic errors. Details of the inner working and/or of the error rules contained in this error grammar are given in neither article. However, Ruth Sanders is aware of the inherent problems linked to the potential size of an error grammar and its influence on parsing efficiency (Sanders 1991, p. 77).

Vosse also uses errors rules for some specific structural errors such as "some rare constituent order problems and punctuation problems" (Vosse 1992, p. 114). However, Vosse's system does not have the goal of treating structural errors and this is seen only as a little plus in the detection of orthographical and morpho-syntactic errors.

METAL (Thurmair 1990) employs specific grammar rules, 'fallback rules' to parse ungrammatical sentences. "In this approach, we do not want the fallback rules to fire except if all other rules failed; i.e. we have to avoid that rules which build grammatical structures are not selected, but rules which are meant as fallback rules fire in 'regular' parses" (Thurmair 1990, p. 366). Rules are therefore exploited in a specific order.

'Repairing Errors in Computer Aided Language' (RECALL) also "uses error-grammar rules to handle errors on the syntactic level and idiomatic errors that surface on this level" (Murphy, Krüger, and Grieszl 1998, p. 65) in addition to the grammar rules which define well-formed sentences.

Thus, one can see that error grammars are often used for syntactic checking in CALL systems, either as the main technique or as a subsidiary one.

2.6 Constraint relaxation

Constraint relaxation is probably the most widely used technique for parsing ungrammatical sentences. Thus, there seems to be nearly as many variants to constraint relaxation as there are of systems using this technique. For the time being, we define a constraint as a condition, or a set of conditions, which must be fulfilled in order for some actions of the parsing algorithm to be taken. A more complete definition is given in chapter 3, section 3.1.1. We begin this section by giving a general description of constraint relaxation, explaining the main ideas and the core technique. Details on different variations are given along with the descriptions of particular systems.

2.6.1 Description

The term 'constraint relaxation' presupposes the existence of constraints. These constraints must be part of a grammar which defines which input is considered grammatical and which is not. "All of the techniques follow a general paradigm of relaxation wherein a normative grammar is assumed which describes the set of acceptable inputs to the system" (Kwasny and Sondheimer 1981, p. 101). Linguistic rules form the basis of the normative grammar that the parsing algorithm follows. "This approach rests on the assumption that linguistic rules can be specified, at least in part, in terms of constraints. For example, the rule for constructing a simple declarative English sentence specifies that a subject noun phrase must be followed by a verb phrase, with the constraint that the subject and verb must agree in person and number." (Catt and Hirst 1990, p. 19). The subject–verb agreement condition has to be fulfilled in order for the rule to apply. If the rule applies, then the actions associated with it are taken. If the condition is not met, the actions are blocked and the parse stops.

When a condition is not met, several options are open. One can simply stop the parse at that point, which would only indicate, at best, the location of the error. One can completely ignore unmet conditions, thus allowing the system to parse ungrammatical sentences without warnings, which is far from what one hopes of a CALL system. One can relax the condition, that is, allow the parse to continue, even though the condition is not met, not ignoring the condition but transforming it slightly. This path was taken by Weischedel and Black who "suggest that, instead of forcing all predicates to be satisfied or ignoring the information inherent in them, that the designer should designate that *certain predicates can be relaxed*, with a record being kept of each predicate not satisfied during parsing" (Weischedel and Black 1980, p. 99, emphasis in original).

It is not necessary for all constraints to be relaxable. Indeed, one could classify constraints or predicates in at least two categories. "Absolutely violable predicates can be permitted in cases where the test describes some superficial consistency checking or where the test's failure or success does not have a direct effect on meaning, while conditionally violable predicates apply to predicates which must be relaxed cautiously to avoid loss of meaning" (Kwasny and Sondheimer 1981, p. 102). Loss of meaning is not the only consequence faced by wild relaxation of constraints. Another problem lies in the reduction of the search space through the constraints: once the constraints are relaxed, the search space expands considerably. "Relaxing constraints introduces additional potential for confusion" (Weischedel and Sondheimer 1983, p. 174). "The need to relax the very rules that constrain the search for an interpretation is like opening Pandora's box." (Weischedel and Sondheimer 1983, p. 175).

In order to limit somewhat the extent of the expansion of the search space through constraint relaxation, "only the constraint that seems to be violated is relaxed; all other well-formedness constraints are still effective. Furthermore, the deviance notes record the aspect that deviates from well-formedness, thus allowing pragmatic inferences by later processing" (Weischedel and Sondheimer 1983, p. 164). The record of the unsatisfied predicates is the means to diagnose the kinds of error committed by the users. This is naturally very useful if the parser is working in the CALL area. It is probably a learner module which formats the feedback for the language learner, but this module needs all the information gathered through the records of unsatisfied conditions. Without these records, the system could only detect errors, without informing the users as to the reason behind the ungrammaticality of their productions. While this might be sufficient in some cases of human-machine interface, it is certainly not enough when one is confronted with a computer assisted language learning system.

On a theoretical basis, constraint relaxation should be able to apply to any type of grammar using constraints. Moreover, this technique should offer the possibility to be implemented after the grammar itself. There should also be possible generalizations between relaxations in different grammars. This was the desire of Kwasny and Sondheimer who note that "one of our original hopes for this work was that each of the relaxations would be applicable to a broad set of individual grammars. However, it quickly became obvious that the relaxations and grammar work best if they are developed together " (Kwasny and Sondheimer 1981, p. 101). This is not to say that relaxing constraint after the grammar is in working conditions is not possible. Nevertheless, the work is simplified when the grammar and the relaxable constraints are designed together from the beginning.

2.6.2 Systems using constraint relaxation

No less that fourteen systems using constraint relaxation are mentioned in this section. The amount of space devoted to each of them mostly depends on the relative wealth of the information found about them.

The BKSW systems

BKSW stands for Black, Kwasny, Sondheimer and Weischedel. This abbreviation is used in the current section to describe the related systems described in Weischedel and Black (1980), Kwasny and Sondheimer (1981) and Weischedel and Sondheimer (1983), as no name could be found in the above mentioned articles for this family of grammar checkers.

The BKSW systems were designed both for natural language understanding systems in English with native speaker users and for German with the goal of inclusion into a CALL system. One of the aims of the authors was to demonstrate the constraint relaxation technique which they were among the first to use. Their parser is implemented with augmented transition networks (ATN) with test-bearing arcs. "In ATN terms, the significant blocks that keep inputs with co-occurrence violations from being parsed as the user intended arise from a failure of a test on an arc, or the failure to satisfy an arc type restriction, e.g., the failure of a word to be in the correct category" (Kwasny and Sondheimer 1981, p. 100). In BKSW, a sentence which can traverse the unmodified ATN is grammatical by definition, the ATN encoding only normative grammar rules. These tests can be relaxed to allow parsing of ungrammatical sentences. Original tests on arcs are always tried first, thus ensuring that grammatical sentences are parsed correctly and restraining the use of relaxed constraints only to the tests which would fail otherwise.

BKSW can relax two different types of constraints: 'test relaxation' and 'category relaxation'. The first one is further divided into absolutely or conditionally violable constraints. Absolutely violable constraints can be assimilated to feature coherence checking, while conditionally violable constraints have an effect on the syntactic structure being built. For absolutely violable constraints, "the opposite value of the predicate is substituted for the predicate during relaxation" while for conditionally violable constraints "a substitute predicate is provided" (Kwasny and Sondheimer 1981, p. 101). Subject-verb agreement is given as an example of an absolutely violable constraint, and the distinction transitive-intransitive as a conditionally violable one.

Category relaxation implies the creation of a hierarchy of lexical categories. Whenever an arc fails because a specific category has not been found in the next token of input, the arc predicate is relaxed by looking for an input token of the parent class in the hierarchy. For example, if a demonstrative pronoun is called for and none is found, then the relaxed predicate checks whether the input token is a pronoun, independently of its subcategory (Kwasny and Sondheimer 1981, p. 102).

As BKSW systems treat errors through several techniques beside constraint relaxation, it is not always obvious to sort which ones are treated by which technique. It seems that errors of co-occurrence (including agreement and preposition selection), and word subcategory are dealt with through constraint relaxation. A list of eleven treated phenomena is given in Weischedel and Sondheimer (1983, p. 164) but it is not clear if all are the result of constraint relaxation. The list is reproduced below.

- failed grammatical tests
- word confusions
- spelling errors
- unknown words

- restarted sentences
- resumptive pronouns and noun phrases
- contextual ellipses
- selection restriction violation
- metonymy
- personification
- presupposition failure

Obviously, not all item of this list are actual errors. Some are rather structures which parsers are known to have difficulty with. Moreover, some do not concern grammar at all. As often, success rates for the BKSW systems are not given.

The EPISTLE/CRITIQUE system

The EPISTLE system, developed by IBM, is "intended to provide office workers with intelligent applications for the processing of natural language text, particularly business correspondence" (Miller, Heidorn, and Jensen 1981, p. 649). This system is therefore intended for native speakers and not for language learners and even less for the language learning process. The aims of this system are much broader than error detection but we will narrow our description of the system to that aspect.

"The EPISTLE's parser is written in the NLP programming language, which works with augmented phrase structure rules and with attribute-value records, which are manipulated by the rules. When NLP is used to parse natural language text, the records describe constituents, and the rules put these constituents together to form even larger constituent (or record) structures" (Jensen, Heidorn, Miller, and Ravin 1983, p. 148).[4] The parsing algorithm is left-to right, bottom-up, with parallel processing. The authors divide the parser into three components: the core grammar, containing about 300 syntax rules; peripheral procedures resolving ambiguity by ranking alternatives; and fitting procedures to handle parse failures.

"The EPISTLE system addresses only the tasks of grammar and style checking of texts written in English. Grammar checking deals with such errors as lack of number agreement between subject and verb; style checking points out such problems are overly complex sentences" (Heidorn, Jensen, Miller, Byrd, and Chodorow 1982, p. 305–306). In the domain of grammatical errors, five error categories are treated: subject-verb disagreement;

[4]The NLP language was later renamed PLNLP and complete references to that approach can be found in (Jensen, Heidorn, and Richardson 1993).

wrong pronoun case; noun-modifier disagreement; non-standard verb forms; and non parallel structures as in example (10) taken from Heidorn et al. (1982, p. 320).

(10) We will accept the funds, send receipts to the payers, and *crediting* their accounts at the same time (should be *credit*).

The EPISTLE system has a three step algorithm to detect grammatical errors. The sentence is first parsed "using fully grammatical rules (where 'fully grammatical' includes conditions on, for example, number agreement between subject and verb)" (Heidorn et al. 1982, p. 320). If the sentence does not parse, then some constraints are relaxed, some additional rules are included, and the sentence is parsed again. If the sentence parses the second time, then a note is made of the conditions relaxed for the parse to succeed and of the location of the problem in the sentence. Relaxed rules are similar to unrelaxed ones, except that one part of the condition must be negated in the relaxed version and error messages added to the action section of the rule. Some of these rules are very specific, such as the one described in Miller, Heidorn, and Jensen (1981, p. 654), distinguishing between 'who' and 'whom' in relative clauses.[5]

As mentioned above, the EPISTLE system uses other techniques beside constraint relaxation. Error diagnosis and correction for about 15% of the errors only, called 'true errors' in (Harriehausen-Mühlbauer 1991, p. 269), are based on constraint relaxation in the CRITIQUE version of the EPISTLE system. We will not detail them, but specific error rules are added to the rule set to detect some grammatical and style errors. Moreover, a 'fitting procedure' (Jensen et al. 1983) is used for the treatment of fragments and difficult cases of ellipsis, among others.

Available results show 64% of successful parses for a sample of 411 business letters covering 2254 sentences (Heidorn et al. 1982). Grammatical and/or style error detection on a corpus of ten essays written by ESL students produced 63 critiques and 54% of correct ones (Richardson and Braden-Harder 1993). However, it is not explicitly said whether the correct critique percentage refers to the number of errors in the essays or to the number of critiques produced by the system, which does make a difference for accuracy. The numbers provided are not complete enough for the reader to form an good idea of the system's performances.

[5]One should note that in Miller et al. (1981), relaxed rules were manually added to the rule set. Plans were made for an automatization of the relaxation, which seems to be accomplished in later versions of the project, as described in Heidorn et al. (1982) and Jensen et al. (1983).

The BRIDGE and MILT systems

The U.S. Army Research Institute (ARI) BRIDGE and MILT systems are of the same family, MILT being developed as the successor of BRIDGE. Both BRIDGE and MILT are full CALL systems of which the NLP component and error detection tool is only a facet. Both have been described extensively (Holland, Maisano, Alderks, and Martin 1993, Holland 1994, Sams 1995, Weinberg, Garman, Martin, and Merlo 1995, Kreyer and Criswell 1995, Kaplan and Holland 1995). We will set aside most of the CALL system to concentrate on the parser and its error diagnosis technique, developed for ARI at the University of Maryland, USA.

The parser used in the BRIDGE and MILT systems follows a principle-based design relying on the government and binding theory developed by Chomsky (1981). This framework "provides construction-independent formulas grouped into interacting modules that can be parameterized" (Weinberg et al. 1995, p. 25). The setting of those parameters differentiates languages. The parser is built with principles fixed for all languages and parameters that can be set individually for different languages. In the BRIDGE system, the parser was developed for both German and Arabic, thus demonstrating its portability to other languages and the usefulness of the principles and parameters framework. For the MILT project, Spanish was added to the set of languages. The parser uses a modified shift-reduce algorithm and is implemented in Prolog.

Although the term is not explicitly mentioned, the BRIDGE/MILT parser employs a form of constraint relaxation. "The use of conditions has several advantages. (...) It is flexible because it separates those features that guide the parser from the features that annotate the tree. (...) this property of our system is crucial for error tolerance and error detection" (Weinberg et al. 1995, p. 38). In fact, none of the conditions governing the features for tree annotation seem to be required to be true. In a way, the parser operates on a completely relaxed mode and only certain types of errors are detected, the others being ignored. "Switches, called flags, determine how restrictive the parser will be in detecting errors. If all the possible flags are set, the parser runs in its most restrictive version; otherwise, it recovers from mistakes whose flags are not set through a mechanism of producing default values on failure of feature matching" (Weinberg et al. 1995, p. 42). None of the tree annotating conditions being ever enforced, it is equivalent to having all the constraints relaxed at the same time. This is feasible because none of the conditions touches the fundamental structure of the sentence representation being built.

The error types taken into account in the BRIDGE system for German are "subject-verb agreement (person, number, gender); case of sentence subject; article-noun agreement (gender, number, case); preposition-noun agreement; verb-preposition agreement; word order errors: position of finite verb in main clause (verb second), position of finite verb in subordinate clause (verb final); auxiliary selection in compound tense (*haben* vs. *sein*)" (Weinberg et al. 1995, p. 39–40, emphasis in original). As one can see, the error types cover mostly agreement features, which are very often treated with constraint relaxation as their violation does not change the structure of the parse tree.

Performances are given only for the parser used in the BRIDGE tutor. Whether small or extensive changes were made to the parser between the BRIDGE and the MILT tutors is unknown to us. The parser was tested on 200 made-up German sentences, including "likely constructions and errors of early intermediate students. The parser appropriately parsed 135 out of 141 (96.4%) correct sentences and 61 out of 64 (95.3%) incorrect sentences (that is, generated the appropriate error messages for these examples)" (Weinberg et al. 1995, p. 43). These results seem excellent. One must note, however, that the parser was not given real language learner input. Moreover, nothing is known on the system's performance when more than one error is encountered in a sentence, as is often the case with real data. As Weinberg et al. (1995) themselves recognize it, more testing would be needed. Unfortunately, we have not encountered the results of later testing of the parser in our readings.

Paragram

Paragram is not especially designed for language learners, although it "will accept sentences which do not fit the grammar, while noting in which ways the sentences are deviant" (Charniak 1983, p. 117). Paragram is a deterministic parser based on Marcus' parser Parsifal (Marcus 1980). In Parsifal, rules were grouped into packets and ranked according to a priority score. The rules of active packets were tried in their order of priority. "Rules in Parsifal are of the typical situation/action type" (Charniak 1983, p. 120). Paragram differs from Parsifal in being able to account for ungrammatical sentences, which implies some modifications of the parser.

Instead of trying the rules by their order of priority and taking the first possible one, Paragram tests all the rules of an active packet one after the other, returning for each rule a 'goodness rating'. "The goodness rating of a rule is the sum of the values returned by the rule's atomic tests. Each atomic test will add to the score if it succeeds, and subtract if not" (Charniak 1983, p. 123). Only the rule with the highest rate is then applied, thus maintaining the deterministic nature of the parser. When encountering an ungrammaticality, all the goodness rates are rather low as not all the atomic tests succeed in any rule, but there is still a rule bearing the best rate. "Furthermore, Pragram can tell the user where the sentence is ungrammatical, since it is only with ungrammatical sentences that the values of the scores go below zero. It is also possible that Paragram can tell 'why' the sentence is ungrammatical, if one makes the reasonable assumption that the sentence is ungrammatical 'because' the rule which succeeded, but with a low score, should have been matched exactly" (Charniak 1983, p. 124–125).

Charniak's technique for robust parsing can be assimilated to constraint relaxation. Indeed, constraint relaxation is allowing a rule to apply although its conditions are not fulfilled. In Paragram, all the rules receive a score depending on the conditions which are fulfilled and those which are not. The rule with the best score is applied, but some of the atomic tests of this rule might have failed. Thus, Charniak allows rules to apply even if the rules' conditions are not met. Moreover, the scoring system, adding a small bonus when an atomic test succeeds and subtracting a heavy penalty when it fails, ensures that only the rule which violates the fewest conditions will be applied. Thus, constraints are bypassed only when absolutely necessary.

Full results of tests of Paragram are not given. We know, however, that Paragram does not handle wh-questions and that there are some classes of ungrammaticality which are not treated, such as the use of a modal instead of an auxiliary, word repetition, and scrambling. Unfortunately, we do not know Paragram's success rate on ungrammatical sentences in terms of completed parse and error detection/diagnosis.

Grammar-Debugger

Grammar-Debugger (Chen and Xu 1990) is a parser designed for Chinese learners of English as a foreign language (EFL). Also derived from Parsifal (Marcus 1980), it uses a different variation of constraint relaxation than Charniak's Paragram. Chen and Xu keep the priority system of Parsifal but split the rule conditions into two parts. "Primary conditions are compulsory. They determine the nature of the rule. Each grammar rule can be entered only when its primary conditions are satisfied. Secondary conditions are optional. They test the well-formedness of the entered input. If they are satisfied, the input is grammatical. The grammar rule can be passed legally. If they are not satisfied, the input is ungrammatical. The grammar rule is passed illegally, and a specific error message along with the words which cause the mistake and the expected correct form are printed" (Chen and Xu 1990, p. 67).

Primary conditions determine the shape of the syntactic tree obtained from the parse, while secondary conditions operate on features. The choice of a rule depends only on its priority and on the satisfaction of the primary condition. Once a rule is selected, it applies whether the secondary conditions are met or not. Secondary or optional conditions are in fact relaxed constraints. Their non-satisfaction directly triggers error messages indicating the type and location of the mistakes.

Grammar-Debugger can handle errors of several types:

- subject-verb agreement, also in relative sentences;

- determiner selection, including for mass nouns;
- and verb mode (past participle, continuous, etc.).

As often, there is no indication of the success rate of the parser on either grammatical or ungrammatical sentences. Thus, it is impossible to assess the actual performances of this system.

Menzel and Schröder's system

Menzel and Schröder's system (MS) is described in several articles, linked to error diagnosis and CALL systems (Menzel and Schröder 1998a, Menzel and Schröder 1998b) or not (Heinecke, Kunze, Menzel, and Schröder 1998). They propose a robust parsing system using an eliminative parsing mechanism and graded constraints. "An eliminative approach starts with a totally ambiguous representation containing all possible dependency structures of an utterance at the same time. The parser then tries to discard local interpretations from this structure by applying unary or binary constraints on dependency relations until, — hopefully — a single global reading survives" (Menzel and Schröder 1998b, p. 47). This has been equated to a (partial) constraint satisfaction problem.

The MS's system makes "use of graded, hence defeasible constraint on all levels" (Menzel and Schröder 1998a, p. 486). Thus, it is akin to constraint relaxation. The choice of the final structure depends on a ranking based on the scores (between 1 and 0) assigned to each constraint. The final score is the multiplication of all the constraints that have been violated to reach the particular structure. Not all constraints have to be satisfied in order to obtain a complete structure for a given sentence. The score assigned to each constraint is an indication of its importance. Hard constraints have a score of 0 and are used to bar completely ungrammatical structures. A low score, but above 0, indicates a well-formedness condition such as number agreement. Scores close to 1 are used to indicate stylistic preferences instead of actual errors.

Tests were conducted on a single sentence which "has been distorted by introducing different kinds of syntactic errors" (Menzel and Schröder 1998a, p. 490). The results are good when the input sentence contains zero or one error, the sentence is then always analyzed correctly. With two errors in the same sentence, the correct analysis is found in 90% of the cases. This success rate drops to 60% for 3 errors and is at 0% for five or more errors in the short sentence that served for the experiment. Unfortunately, it seems that no test was conducted with actual data and in particular with learner sentences.

Miniprof

Miniprof's name was appropriately chosen, as Miniprof's purpose, grammar, and lexicon are all actually rather limited. The lexicon contains roughly 100 words (Labrie and Singh 1991, p. 13) and the grammar, listed in figure 4 of (Labrie and Singh 1991, p. 14), contains exactly 14 rules. Some of the rules are augmented with conditions, for number, person and gender agreement, and for elision of the negative particle 'ne'.

The parser, following a top-down algorithm, "tries to parse as much of the sentence as possible and identify errors" (Labrie and Singh 1991, p. 15). If during the parsing process a rule matches but not its additional condition on agreement or elision, the sentence is recognized but the error is flagged. Skipping an element from the rule, such as the negative 'pas' is also possible, which again triggers an error flag. Thus, Miniprof's parsing is akin to constraint relaxation. If what is needed is there, the parse proceeds normally, otherwise it recovers by inserting an error flag for later use and parsing resumes.

The parser is able to detect 12 error types under the authors' typology (see figure 5 in Labrie and Singh (1991, p. 19)). A 'tutor' component is in charge of detecting errors of context or linked to the exercise questions. Results are not given for this toy system. Plans for expansion, in particular to treat spelling errors and word order errors are mentioned at the very end of the paper, but it is not said if this would be achieved with the same diagnosis technique.

Vosse's Dutch Checker

Vosse's parser uses a shift-reduce algorithm with an augmented context-free grammar (ACFG). "Simply put, an ACFG is a Context-Free Grammar where each non-terminal symbol has a (finite) sequence of attributes, each of which can have a set of a finite number of symbols as its value" (Vosse 1992, p. 113). Without naming the technique, Vosse's system uses constraint relaxation by not failing when the added attributes do no match. The system marks the non-matching attributes and the parse proceeds. Once parsing is over, the marks are passed on to the correction part of the system.

Vosse uses this technique only to detect morpho-syntactic disagreement. His system is designed for spelling and morpho-syntactic errors, not for other types of errors, although some structural errors can also be detected by the adjunction of error rules.

Numerous experiments were carried out to test the results of this system. They are reported in Vosse (1992). To summarize them, it appears that the system has no major difficulty in detecting the errors it is designed for on all kinds of corpora. However, parsing time is problematic, especially on real texts when the number of words per sentence increases. For this reason, the system is designed to work in batch mode rather than interactively, which would not be ideal for language learners.

Schwind's error analysis and explanation system

Schwind (1988) compares her error analysis and explanation system (EAES) (Schwind 1986, Schwind 1988, Schwind 1995, see also 2.5.2) to Menzel's and Weischedel's treatment of ill-formedness and states that "Weischedel's treatment of syntactic and agreement errors is very similar" (Schwind 1988, p. 608) to what she does. EAES is built with a feature grammar where "the result of unification is the unified elements and the set of the pairs of elements for which the unification did not work out" (Schwind 1988, p. 609). The structure of the sentence is still created through production or transformation rules to which operations on the features of the participating elements are added. The modification of the unification procedure allows Schwind's system to continue building a syntactic tree although all the features do not necessarily match. There is a dissociation of parsing and feature checking processes. Apparently, the tree is build on the basis of the lexical category of the elements. By analyzing the unification sites where unification did not succeed and in looking at the sets of non-unified pairs of features, the system derives the most plausible error explanations.

Other systems

Some other articles mention systems which use constraint relaxation for grammar checking. However, the information gathered about these systems is not worth separate descriptions and they are grouped here.

Scripsi is a full CALL system prototype described in Catt (1988). It includes a student model which takes advantage of the knowledge of the L1 of the language learners. Part of the error diagnosis system uses constraint relaxation: "*Scripsi* detects such overgeneralization by means of *constraint relaxation*; that is, by suspending constraints when they are not observed" (Catt and Hirst 1990, p. 19, emphasis in original). Errors diagnosed by constraint relaxation include mistakes of subject-verb agreement, subject inversion, and use of do-auxiliary in questions. The particular variant of the constraint relaxation technique is not described in the above cited article.

Kempen (1992) describes a system which uses constraint relaxation for diagnosing agreement errors. The parser is an LR(1) of the shift-reduce family with terminal symbols augmented by feature matrices. During the reduce action, a unification procedure is applied to the feature matrices. "Unification succeeds not only in case all features match, but also when there are feature violations ('constraint relaxation'). In the latter case, a diagnostic flag is attached to the resulting subtree" (Kempen 1992, p. 196). The technique is used only for agreement checking and no results are given in the above cited article.

The XTRA-TE system of Chen and Barry (1989) is described in Chanier, Pengelly, Twindale, and Self (1992) as an intelligent tutor system (ITS) build on top of a Chinese-English translation system and using constraint relaxation to diagnose errors. The system has "a full coverage of subject verb agreement and of pronominal errors. XTRA-TE is able to diagnose others types of errors but leaves aside the problem of incorrect word order which may imply an expansion of the grammar" (Chanier et al. 1992, p. 135).

Schulze (1998)'s Textana system is a German grammar checker for learners of English mother tongue. A grammar based on the Head-Driven Phrase Structure Grammar (HPSG) formalism (Pollard and Sag 1994) and a parser have been implemented in Prolog. It "covers a substantial fragment of German. The dictionary prototype currently contains approximately 500 morphemes (roots, lexical affixes, inflectional affixes, verbal prefixes)" (Schulze 1998, p. 218). The first step was to encode the grammar of standard German with 'hard' constraints. These constraints are then relaxed to enable the system to parse learners' sentences. Whenever a condition on a rule is not fulfilled and there is the possibility to record an error, the rule is still allowed to apply. Selecting which constraints to relax will be decided after the analysis of a set of texts produced by the target population. Unfortunately, results are not given in the cited article.

The Athena Language Learning Project (ALLP) NLP system is described by Felshin (1995). Although the information provided on the diagnosis component of the system is relatively succinct, we learn that "errors in agreement are anticipated by relaxing constraints in the rules that comprise the system" (Felshin 1995, p. 257). Other types of errors are treated with different means. Extensive use of scoring and of penalties for discovered errors is made in the system in order to find the most likely parse for the sentence, keeping strictly in mind the likelihood of a given sentence to be typed in by a language learner of a specific level.

2.7 Other techniques seen in literature

This section features a few other techniques for error diagnosis which were found in the literature. These techniques are not attributed a whole section of their own because there is not enough information known about them.

2.7.1 Delayed checking

Delayed checking is, as a working definition, a technique where a syntactic tree is built first, without feature checking. Once the tree is completed, feature checking procedures apply to check the sentence coherence.

Cornu's lexical functional grammar (LFG) (Bresnan 1982) parser-based checker (Cornu 1994, Cornu 1997) proceeds in three steps. The text is submitted to a syntactic parser which creates syntactic trees. Rules on functional values build functional structures on top of the syntactic trees. Functional analyses are ranked by type and number of found errors. Only the structure with the least number of errors is retained as being the most likely. In other words, uniqueness condition violations are recorded while building c-structures, while "coherence and completeness conditions are only checked after the f-structures have been built completely" (Cornu 1994, p. 186). Feature checking is in fact delayed until the syntactic trees are completed. Some modifications of the parsing algorithm have to be made in order for the LFG parser to detect errors. Most importantly, the unification process has been altered in order not to destroy the information pertinent to error correction. Default values are then set for the values and these defaults are used to decide which non-coherent value to retain. Secondly, Cornu introduces what he call "règles d'environnement lexical" (Cornu 1997, p. 117) to deal with some long distance agreement phenomena and specific errors of non-native speakers, such as wrong case of the pronoun. The system can correct errors of agreement, past participle, subjunctive and preposition for infinitival clauses, adjective-noun word order, verb subcategorization, and contracted determiners. The results show that 70% of the errors of a small corpus are correctly diagnosed. However, Cornu (1997, p. 196–197) concludes that the LFG formalism alone is not really appropriate for error correction. Reasons are the exponential amount of data that the software has to keep in memory; and that adding new rules and new lexical data would create new false positives and non-detection as well as new correct detection.

ReGra is a grammar checker for Brazilian Portuguese designed for native speakers. "The parser utilizes a Context-Free Grammar with relaxation constraints-based techniques for permitting matching of non-grammatical sentences" (Teixeira Martins, Hasegawa, Volpe Nunes, Montilha, and Novais De Oliveira 1998, p.288). It appears that syntactic configurations are planned in advance, the constituents attach to match the input sentence without constraint (such as agreement) checking. Agreement checking is performed once the whole sentence has been parsed and triggers, if necessary, the correction process. The technique used is therefore at the border of constraint relaxation but should more reallistically be classified as a case of delayed checking. A syntactic tree is built on which agreement checking is then performed in order to locate potential errors. No constraint is actually relaxed from a stricter form to allow a parsing process to continue.

2.7.2 Predication-driven parsing

Instead of the usual grammar and parser with its set of rules, using either error-rules or constraint relaxation for diagnosis purposes, DeSmedt (1995) proposes a case-grammar technique he calls 'predication-driven parsing'. "This approach proceeds by first identifying the operative verb and then attempting to interpret the remainder of the input in terms of its complements. Given a transitive verb, for instance, the parser begins looking for a noun phrase that could function as the predications' direct object" (DeSmedt 1995, p. 158). Error diagnosis is carried out by first assigning all possible elements of a sentence to its correct function. If some elements remain, such as noun phrases, as well as empty slots of the verb, it implies that the sentence is not fully grammatical. The NLP system in DeSmedt's 'Herr Komissar' then applies a least distance heuristic in order to determine the best possible match. While in many other systems results are not given, those provided here tend to show that this technique is really quite proficient. "Her Kommissar was able to detect and correct 92.8% of all errors committed by students, with a false correction rate (i.e., detecting an error where none is present) of only 1.1%" (DeSmedt 1995, p. 159). These numbers are so good that one really would like to test the system and make sure for oneself to the accuracy of those results. Perhaps the restrained domain of application designed for the game is partially responsible for these good figures by limiting the kind of sentence structures which were actually used by the test students.

2.7.3 Link Grammars

After closer inspection, the Francophone Stylistic Grammar Checker (FSGC) described by Brehony and Ryan (1994) does not seem to have much to do with style at all. It seems to be purely a morpho-syntactic checker and it is in this frame of mind that we describe it here.

Sentences are parsed using link grammars (Sleator and Temperley 1991), which formalism is briefly described by Brehony and Ryan (1994, p. 258–260). Error detection occurs during a post-processing phase on the links to check some further conditions of grammaticality. "However, if a sentence containing one of the stylistic errors that are to be detected is to survive to post-processing, it must first pass the initial three criteria. This meant that for each error, it was necessary to alter the dictionary definition formulas for relevant words, to allow the sentence containing the error to be accepted as grammatical" (Brehony and Ryan 1994, p. 262). This lead to the creation of new links which could, to take the example given by Brehony and Ryan (1994), link a singular subject to a plural tensed verb. Error detection then proceeds by scanning the links generated by the parser for the added-on links.

The authors give only one type of error as example. They do not mention any type of evaluation of their system. However, they have the honesty to mention some of the problems encountered by this software: the large memory resources needed, the impossibility to choose among several linkages for the same sentence otherwise than randomly for the linkage to undergo post-processing and error-detection, and the fact that "only errors pre-programmed are detected" (Brehony and Ryan 1994, p. 265).

2.7.4 Unclassified

One can find a number of articles on CALL systems in literature which mention error detection but which do not give a precise enough description of the inner workings of the system to allow us to classify it as using one or another diagnosis technique. However, we do not see any reason not to mention them in a chapter which is devoted to existing software.

BAP (Cook 1988) stands for *Basic Parser* as it is programmed in BASIC. It caters to beginners only, working on the assumption that only a very restrained vocabulary and small subpart of the grammar of English is needed for that category of users. It "anticipate[s] the students' possible mistakes" (Cook 1988, p. 63) and stores error reports each time tests fail during during parsing.

Courtin et al. (1991) agreement checker system works on a dependency tree computed by a parser. The checker verifies the dependencies through a number of rules, each containing a header, a condition, and an action. To be applicable, the rule header must match the pair of nodes of the dependency being processed. If the conditions are then met, the action is taken. Otherwise, an error is detected and the information is passed on to the correction module which will make suggestions to replace the erroneous word. Other error detections beside agreement are planned but not described in the cited article.

Schuster (1986) proposes a system called VP^2 which compares parse trees from a prestored answer and the learner's answer. Mismatch between the two trees triggers error messages and explanations. The grammar used for parsing is the grammar of the target language, of the first language, or a mix of the two. VP^2 treats only errors of construction for phrasal verbs and verbs requiring specific prepositions.

Nagata (2002) uses an error detector in the BANZAI CALL software which checks the coherence of the learner input compared to a single target answer. The error detector itself runs on parsed versions of both the target answer and the learner input. Precise details as to how the comparison is actually performed are lacking.

2.8 Commercialized checkers

Most grammar checkers developed as scientific research projects never reach the commercialization stage. An exception is Vosse's system which is described in 1992 to be "in its final testing phase" (Vosse 1992, p. 111) and planned for commercialization in the same year. Another difficulty regarding commercial checkers issued from research is that their name might have been changed for advertising purposes which renders tracing research project to off-the-shelf software a difficult task.

This section reports on the little information that was gathered on the techniques used by commercial grammar checkers and on published and evaluated performance of a small number of commercialized grammar checker for French.

2.8.1 Techniques used

Getting information on the diagnosis techniques used by commercial checkers is not an easy task as this information is not readily available. However, pieces of information crop up from time to time. While describing the potentiality of adapting a commercial checker for use with learners of English as a second language, Brock presents *Grammatik IV* which uses two diagnosis techniques, the first one being pattern matching. “The second programming technique employed by *Grammatik IV* in analyzing text and providing the writer advice is parsing rules” (Brock 1990, p. 54). This, unfortunately, says nothing about the techniques used today as the just cited article was published more than ten years ago and reports on a system by necessity a little older.

The only recent information found on the inner workings of a grammar checker is the following: “Like most commercial grammar checkers, NativeEnglish (like its parent CorrectText®) Grammar Correction system) *identifies errors by matching patterns of known errors in its database*” (Hu, Hopkins, and Phinney 1998, p. 96, emphasis in original). We do not know, however, on what authority or research Hu et al. base themselves when they generalize to other systems.

2.8.2 Published and evaluated performances

The performances of monolingual grammar checkers are often disappointing when used with language learners’ texts. “Nous constatons que les correcteurs monolingues commerciaux détectent au maximum une erreur sur deux et produisent un nombre de fausses détections relativement élevé” (Cornu 1997, p. 80). Considering the lack of bilingual grammar checkers, monolingual systems are still often used in language learning circumstances. Evaluations for some of the systems have been available at least for the last ten years in scientific journals (Granger and Meunier 1984), conference proceedings (Davies and Wei 1997), and the computer science press (Dinnematin and Sanz 1990, Eglowstein 1991, Sanz 1992). We will restrict our evaluation to recent versions of grammar checkers for French. In chapter 5, section 5.3, we will compare a commercial checker to our own error diagnosis system.

‘Le Correcteur 101’, of Machina Sapiens, was for a long time the leader in

French grammar checking. This product is evaluated in (Mogilevski 1998). Although "Le Correcteur attempts to provide a complete grammar analysis of any given sentence" (Mogilevski 1998, p. 184), "some syntactic mistakes prevent the program from completely analyzing sentences. In this case, partial analysis is given, and users can obtain indications of how to modify the segments of the sentence by double clicking on them" (Mogilevski 1998, p. 185). The parsing technique used is unknown but it apparently contains an impressive number of rules as "the new version of Le Correcteur contains thousands of grammar rules that were not included in the previous version" (Mogilevski 1998, p. 187). Although some kind of parsing must be done in order to obtain a sentence analysis, error detection is possibly achieved with a very large number of very specific error rules. If this is indeed the case, then errors are not treated in a generic way and detecting new types of errors will involve the design of numerous new rules. One must admit, however, that, as can be seen in Tables 2.1 and 2.2, Le Correcteur's results are very good.

Burston (1998) evaluated 'Antidote 98', of Druide Informatique, and compared it to 'Le Correcteur 101', its most direct competitor. This system also offers a complete grammatical analysis of sentences and partial analysis whenever it cannot analyze the whole structure. No information is provided as to the inner workings of Antidote 98, so no suppositions can be made on its error diagnosis technique. "Le Correcteur and Antidote were tested against 50 second-year advanced student essays containing 1,262 morphosyntactic errors" (Burston 1998, 209). The results are reproduced in Table 2.1.

Table 2.1: Comparison between Le Correcteur and Antidote

Program	Number of Errors	Errors Corrected	Errors Flagged	Total	Misid-entified	Unde-tected	False
Correcteur	1,262	74%	17%	91%	7%	2%	49
Antidote	1,262	77%	5%	82%	11%	7%	28

The results on morpho-syntactic errors are very good. Up to 98% overall of detected errors with Le Correcteur 101, and 93% for Antidote. The number of false detections, although not null, is also relatively low. However, only morpho-syntactic errors were included in this test, comprising agreement errors, verbal conjugation and spelling. Nothing is said on the capability of these two grammar checkers to detect errors of syntax such as verb complementation or word ordering, difficult points for language learners.

Granger, Meunier, Verhulst, and Watrin (2001) compared three French grammar checkers: Cordial 7 Pro (version 7.00, Synapse); Correcteur 101 Pro (version 4, Machina Sapiens); and Antidote (edition 2000, Druide Informatique). The comparison was centered on the detection and correction of number agreement errors on a corpus of thirty-five made-up sentences (corpus A) and on a corpus of sixty authentic sentences produced by learners of French (corpus B), of which nine contained a referential error. The results for error detection are given in Table 2.2 (Granger et al. 2001, p. 18).

Table 2.2: Comparison of three grammar checkers

	Cordial	Correcteur	Antidote
Corpus A	74%	85.5%	97.1%
Corpus B	47%	60%	67%
Corpus B without referential errors	55%	70.5%	78.4%

Unsurprisingly, none of the checkers was able to detect referential errors. Results for made-up sentences are significantly better than on authentic sentences. This is certainly due in part to the shorter length of made-up sentences and to their simpler structure. One should also note that the authentic sentences contained other errors besides number agreement errors, which complexified the task. These results undoubtedly put Antidote in first position. The authors have proposed no supposition as to why some errors were detected and others were not or as to the inner workings of these commercial checkers.

These recent comparisons prove that, in the space of a couple of years, the ranking of grammar checkers can be inverted.

2.9 Summary and conclusion

There exist many CALL systems which include a grammar checker at the prototype stage, as can be attested by the numerous literature cited in this chapter. It is, however, not always easy to have relevant information on the inner workings of the grammar checkers described. One can see at least three reasons for this. The first one is that the focus of many articles is not on the grammar checker per se, but on the whole system. It might also describe the results obtained by the checker without necessarily describing it in depth, but making references to unavailable literature. Secondly, the authors do not always have the space to give as many details as they would like to. Finally, some authors might prefer not to give too many implementation details, especially if they intend to commercialize their product, so as to keep the competition at bay.

Leaving aside pattern matching which does not involve parsing, two of the techniques described in this chapter stand out of the lot as the ones used most widely in the CALL community: constraint relaxation and error grammars. The wealth of information gathered on these two techniques is much larger than on the others and the shear number of systems using either of these techniques is impressive. However, most of the systems described are only at the prototype stage. Some are even designed at so small a scale that one cannot believe that they were ever destined to grow up to the size needed for commercial use. This raises the problem of scalability: How would the prototypes, and their diagnosis techniques, react to an increase in coverage, in size of the rule set, and in size of the lexicon? Negative answers to this question may be partly responsible for the apparent disappearance of many systems for which one or two articles only seem to have ever been written.

One must also note that many systems combine different techniques, such as adding a few error rules for anticipated errors as well as relaxing some constraints (Felshin 1995, p. 257). Cornu also concludes his dissertation by postulating that combining his LFG checker and the automata technique should produce better results that either one alone (Cornu 1997, p. 287). Vosse adds some error rules for the detection of specific structural errors to his mechanism for morpho-syntactic error detection (Vosse 1992). "Probably large ICALI systems will use a combination of techniques for responding to erroneous input. For example, a parser could combine an error grammar for some typical errors with simple detection for errors not codified in the system" (Sanders and Sanders 1989, p. 18). Combining techniques also shows how some techniques are specialized to treat particular types of error. Using the best possible technique for each particular error type, for example constraint relaxation for agreement errors and error rules for word order errors, should provide better results than using either one, at the same time probably simplifying the implementation of each particular technique. Combining techniques, thus, seems to be the way forward for improvement of grammar checkers. This creates new challenges which have not been discussed before, as far as we now. They are (i) selecting appropriate techniques for combined use and (ii) managing the results of the combined techniques in a coherent manner.

Nothing much seems to have changed over the last two decades in the realm of NLP-based CALL, in fact. CALL systems offering error diagnosis functionalities are not frequent and the use of parsing techniques is even rarer. Not much innovation seems to have been conducted with regards to possible error diagnosis techniques. There seems to be as many systems using constraint relaxation and error rules dating from the late 1990's as from the early 1980's. On the other hand, commercialized grammar checkers for native speakers have made their appearance and improved drastically during the same period. The techniques they use may still be the same as twenty years ago, but the number of person/year of work put into them shows results. The huge amount of work it requires is not often available in research and academic institutions.

Finally, all the error diagnosis systems which have been mentioned in this chapter seem to have been built at the same time as the parser they rely on. From a conceptual standpoint, it makes a lot of sense to do both together. It is indeed easier to structure error rules at the same time as the rules for the correct grammar, for example, than to do it afterwards. However, having to work in two different directions explains perhaps partly the lack of coverage found in many of the diagnosis systems described. Part of the effort is diverted from the diagnosis to the parser itself, and thus each of the two aspects cannot be pushed as far as if it were the only one.

As it is detailed in the next chapters, we take a different approach. Indeed, our starting point is an existing syntactic parser for French with a broad coverage of both the French language and the lexicon. The challenge is thus of a different sort. Instead of creating something brand new, we must adapt a parser which was built for the syntactic analysis of grammatical French sentences, into a syntactic error diagnosis tool for second language learners of an advanced level. The parser and its lexicon bring us the needed coverage in terms of grammar and lexicon. The difficulty then resides in diagnosing as many error types as possible, with a great degree of accuracy, without diminishing the capacities of the parser. From the experience learned in studying other diagnosis systems, we take the option of combining different error diagnosis techniques. Additional challenges are found in the diagnosis technique selection process, as well as in the management their (possibly diverging) results.

Chapter 3

Theory

This chapter is devoted to theoretical considerations concerning three different techniques for error diagnosis in the CALL context. These three techniques are constraint relaxation (3.1), which has already been discussed in chapter 2 in the light of existing systems using it; phonological reinterpretation (3.2); and chunk reinterpretation (3.3). Each of these techniques is described in this chapter from a theoretical standpoint. Section 3.4 investigates distinct possibilities to retrieve the most likely analysis when several analyses are proposed by the system, given the particular context of use. Using several diagnosis techniques implies a way to combine the techniques and the results they provide. Section 3.5 explores three different possibilities for managing these results coming from different sources. Section 3.6 concludes this chapter.

3.1 Constraint relaxation

This section describes the constraint relaxation technique and provides some information on the elements involved in constraint relaxation. A working definition of constraints is proposed (3.1.1), followed by a short typology of different types of constraints (3.1.2), and by an explanation of what is understood by constraint violation (3.1.3). Prerequisites for constraint relaxation are stated (3.1.4), the constraint relaxation technique is explained (3.1.5) and an algorithm is proposed (3.1.6). Deciding which constraints to relax is next explored (3.1.7). Advantages (3.1.8), problematic issues (3.1.9), and variants to the technique (3.1.10) are discussed. A few remarks (3.1.11) conclude the section.

3.1.1 Constraint definition

Before we embark on a description of constraint relaxation, a definition of what a constraint is should first be given. As a first approximation, a constraint is a condition which must be fulfilled in order for the parsing algorithm to take a certain action. This condition can be either simple or complex, that is, made of several sub-conditions connected by logical operators. However, if we take a complex condition, a constraint can either be the whole condition or a fragment of it, and this fragment needs not be reduced to the level of an atomic sub-condition. Let us take a simplified example as an illustration:

(11)a. $[_{\mathtt{DP}}$ The cat $]$

b. $[_{\overline{\mathtt{T}}}\ [_{\mathtt{VP}}$ sleeps $]\ ]$

c. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}$ The cat $]\ [_{\overline{\mathtt{T}}}\ [_{\mathtt{VP}}$ sleeps $]\ ]\ ]$

(11a) can combine with (11b) to form the sentence (11c) only if the complex condition (12) is true.[1]

(12) (11a) must be a DP & (11b) must be a $\overline{\mathrm{T}}$ & (11a) and (11b) must agree in number & (11a) and (11b) must agree in person.

Condition (12) can be divided into four atomic sub-conditions which all need to be true to enable the formation of (11c). Condition (12) is however made up of only two or three constraints. The first two atomic sub-conditions combine into the first constraint (13), which we could name 'category constraint'.

(13) (11a) must be a DP & (11b) must be a $\overline{\mathrm{T}}$

The two remaining atomic sub-conditions can form a second constraint, which we could call 'subject-verb agreement constraint', or each form a constraint of its own, respectively 'number agreement constraint' and 'person agreement constraint'. Thus, a constraint depends on the exact level of granularity one wants to achieve. Independently of granularity, one can define a constraint as being a condition, or a coherent part of a complex condition, on which actions of the parsing algorithm depend.

3.1.2 Constraint typology

Let us schematize a rewrite rule of a grammar as:

[1] Example (12) is a simplified version which does not include all the conditions which would be present in a real system (the information that $\overline{\mathrm{T}}$, as a main clause, must be tensed is missing, for example).

(14) A $\Rightarrow$ B C, constraint set

which can be read as B followed by C combine to form A, as long as each constraint in the set of constraints is fulfilled. The values of A, B, and C combine to form the category constraint. The constraint set contains all the other constraints applying to that particular rule. Applying (14) to example (11), one obtains the following:

(15) TP $\Rightarrow$ DP $\overline{\mathrm{T}}$, {number agreement constraint, person agreement constraint}

Thus, DP and $\overline{\mathrm{T}}$ can combine into a TP only if the number and person agreement constraints are met. If it is not the case, the sentence is ungrammatical. The number agreement constraint states that DP and $\overline{\mathrm{T}}$ must share the same number value (either singular or plural), while the person agreement constraint states that DP and $\overline{\mathrm{T}}$ must share the same person value (either first, second, or third). Example (15) is very schematic and one can think of many more constraints which should accompany this rule. Some of them are linked to the nature of the inner elements of DP and $\overline{\mathrm{T}}$. For example, some verbs, like 'sleep', require an animate or human subject. This kind of constraint, at the border between syntax and semantics, demands that the verb bears a particular feature. If it does not, the constraint does not apply at all, which is different from failing or not succeeding.

(16)a. * $[_{\texttt{TP}}$ $[_{\texttt{AdvP}}$ Very] $[_{\overline{\texttt{T}}}$ $[_{\texttt{VP}}$ sleeps]]]

b. * $[_{\texttt{TP}}$ $[_{\texttt{DP}}$ The cat] $[_{\overline{\texttt{T}}}$ $[_{\texttt{VP}}$ sleep]]]

c. ? $[_{\texttt{TP}}$ $[_{\texttt{DP}}$ The desk] $[_{\overline{\texttt{T}}}$ $[_{\texttt{VP}}$ sleeps]]]

We can thus distinguish at least three types of constraints. First, there are constraints which are embedded in the rule descriptions themselves, such as the category constraint. We call them 'hard' constraints because sentences are highly ungrammatical when a constraint of this type is not respected (16a). The second type of constraints applies to all instances of a given rule type. Agreement constraints belong to this type. We refer to them as 'soft' constraints as sentences which do not obey these constraints are ungrammatical to a lower degree, as in example (16b). Thirdly, some constraints apply only when some prior condition is met. Typically they make use of semantic information or of categorial sub-types (proper noun vs. common noun). We name these 'conditional' constraints. (16c) offers us an example of an unsatisfied conditional constraint: the verb 'sleep' requires an animate subject. (16c) is thus not readily acceptable.

3.1.3 Constraint violation

While parsing a grammatical sentence with a non deterministic parser, many constraints are checked time and time again and some constraints are bound to be violated some time or other. A violated constraint then indicates that the chosen path is not the right one and the algorithm proceeds by selecting another possible route. If all alternatives are tried without success, then the parse fails. With a deterministic parser, the first violated constraint indicates a parse failure, as no backtracking or parallel processing is possible with "strictly deterministic" parsers (Marcus 1980, Marcus 1987). For both types of parsers, a parse failure implies that the sentence is not grammatical.[2] A successful parse indicates a grammatical sentence for which a syntactic structure can be provided.

There are, however, degrees in grammaticality (Chomsky 1964) for which the constraint typology proposed in section 3.1.2 may account. This is what Kwasny and Sondheimer (1981, p. 102) recognize in their differentiation between absolutely and conditionally violable predicates, and Holan, Kuboň, and Plátek (1997, p. 152) between soft and hard constraints. The category constraint seems more important for grammaticality than the agreement constraints, as shown by the dichotomy in example (16) where (16a) has a much lower degree of grammaticality than (16b). (16c) is odd without context, but quite grammatical.

Thus a strong ungrammaticality comes from the violation of a hard constraint (in example (16a), the category constraint), while a minor one comes from the violation of a soft(er) constraint (in example (16b), the person agreement constraint). The violation of conditional constraints often requires a specific context for sentence interpretation (in example (16c), the animate subject constraint).

Constraints are violated whenever an ungrammatical sentence is produced. Several constraints can be violated within the same ungrammatical sentence. The degree of grammaticality of the produced sentence depends on the type and number of violated constraints. To our knowledge, no exact ordering of constraints has ever been done formally. Ranking sentences by their degree of grammaticality could however provide us with precious information about the ranking of constraints, as long as one is able to link specific constraints to each ungrammatical sentence.

3.1.4 Prerequisites for constraint relaxation

As a prerequisite for constraint relaxation, one needs a constraint-based syntactic parser. In ideal conditions, this parser should cover the grammar of

[2]It could also mean that the sentence lies outside the bounds of the parser's grammar. However, we assume in the remainder of this chapter that the parser covers the whole grammar of the language under treatment.

the language exhaustively and exclusively. This means that all grammatical sentences should be able to parse completely and that no ungrammatical sentence should result in a full parse. The unmodified parser should serve the role of a recognizer.

By constraint-based parser, we understand a parser that relies on constraints and on their truth value in order to decide whether to continue along a certain parsing path or not. Each action of the parsing algorithm is linked to one or more constraints. An action is taken only if no constraint is violated.

The grammar used by such a parser needs not be separate from the parsing algorithm, but it must be designed with clearly identifiable constraints. If the constraints cannot be identified one way or another, constraint relaxation cannot be implemented as a syntactic error diagnosis technique.

3.1.5 Error diagnosis by constraint relaxation

Error diagnosis consists in identifying the ungrammaticality of a sentence, in locating this ungrammaticality, and in giving its type. As each site of ungrammaticality is linked to a violated constraint, error diagnosis must identify which constraints are not respected. One possible manner to achieve this is by constraint relaxation.

Basically, constraint relaxation starts with the parser encountering a constraint which fails and blocks parsing. Instead of stopping there or finding an alternative path, constraint relaxation allows the parsing process to override the constraint and to continue. Error diagnosis is achieved by noting which constraint was relaxed and the context in which it was relaxed.

Constraint relaxation implies somehow changing the actions linked to the constraint. The most obvious way these actions are modified is by the insertion of instructions on how to store the location of the relaxed constraint and the type of the constraint. These elements are essential for the diagnosis process as there is, in ideal conditions, a direct link between relaxed constraints and syntactic errors. ‘Diagnosis notes’ are computed every time constraint relaxation is used for error diagnosis purposes. If the actions controlled by a constraint are not linked to values checked within the constraint, then this constraint can be relaxed by simply ignoring it if it is not met. Other types of action modifications are often used. If the actions resulting from a satisfied constraint involve calculations of new (feature) values based on some values checked within the constraint, these calculations might have to be changed or modified if the constraint remains unsatisfied. Indeed, the planned calculation might not be possible if the starting values are not within a certain range. One can then use default values, or a modified calculation method.

A typical calculation which needs modification is the formation of a new value set, which cannot be empty, through the intersection of two value sets

(17a). If the two value sets are not compatible, the resulting intersection is empty (17b). As stated above, this is not legal and one cannot use this result. One must therefore use a default value or indicate that, in the case of the constraint being unsatisfied, the calculation must take the union of the value sets instead of their intersection (17c). This insures that the resulting set is never empty.

(17)a. {singular, plural} $\bigcap$ {singular} = {singular}

b. {singular} $\bigcap$ {plural} = { }

c. {singular} $\bigcup$ {plural} = {singular, plural}

Once the parse is completed, one collects the diagnosis notes produced by the relaxation of constraints. These notes are passed on to a student module, or such like module, for display in an appropriate user-friendly format.

3.1.6 Algorithm

The purpose of this section is not to write down a parsing algorithm, but rather an algorithm for constraint relaxation independent of the kind of parser used.

Let us assume that we have a set of constraints on which depends a set of actions. We define C as the set of constraints $\{C_1, C_2, C_3, ..., C_i\}$ and A as the set of actions $\{A_1, A_2, A_3, ..., A_j\}$.

If all the constraint in set C are satisfied, then all the actions in set A proceed as planned. If one, or several, of the constraints in C is not satisfied, then a set of modified actions A' is invoked. The set of modified actions varies according to which constraint or constraints have not been satisfied. An action can be suppressed, modified or added to set A in order to form set A'. Thus A' is potentially the set $\{A'_1, A_3, ..., A_j, A_{j+1}, A_{note}\}$, where A'_1 is a modified version of A_1; A_2 has been removed; A_{j+1} is an added action; and A_{note} is the action pertaining to the creation of a diagnosis note.

- For each relaxable constraint
  - If C then
    - Apply A
  - Else
    - Apply A'

Thus, a constraint is relaxed only if it is not satisfied and the set of alternative actions A' is activated only in case A could not be used. We will see later (section 3.1.9) that this has its importance for the efficiency of parsers containing relaxable constraints.

3.1.7 Deciding which constraints to relax

Deciding which constraints to relax is a complex matter. It obviously depends on the errors one wants to be able to diagnose. Only the constraints linked to those errors need to be relaxed. Thus, identifying a relation between errors and constraints is of the utmost importance.

Usually, hard constraints are not relaxed because relaxing them would allow the parse to construct uninterpretable structures. In a general way, constraints which are relaxed change only slightly or not at all the overall shape of the syntactic structure of the sentence. Allowing hard constraints to be relaxed would completely upset the sentence structure.

Each application using constraint relaxation must therefore conduct a study to determine the errors it wants to be able to diagnose and use this knowledge to define the range of relaxable constraints. “Depending on the task, communication type, and many other factors certain constraints will be singled out for possible relaxation” (Uszkoreit 1991, p. 239). Agreement constraints are a typical example of constraints relaxed very often in systems using constraint relaxation. The choice of constraints depends on the goals of the application, as well as on other diagnosis techniques used in conjunction with constraint relaxation and which would treat some particular error types.

3.1.8 Advantages

One of the important advantages of constraint relaxation for error diagnosis is that it also provides a full parse of the sentence under scrutiny. The user can therefore benefit not only from the error diagnosis, but also from all other results of a normal parse. In the CALL domain, this includes the possibility to examine the sentence structure, and to receive disambiguated lexical information on single words. Being able to reach a complete parse, even if containing errors and having required the use of relaxed constraints, is also an indication that the sentence grammaticality, although not perfect, is not too far off the mark. In a parser with relaxed constraints, an unsuccessful parse indicates the encounter of an unsatisfied unrelaxed constraint. As constraints that are not relaxable are often hard constraints, such an input sentence has probably a high degree of ungrammaticality. One is thus also able to provide a diagnosis of ungrammaticality without being able to indicate the precise nature or location of the error.

Moreover, reaching a complete parse with any parser, and even more so with a parser with relaxed constraints, is an indication that most, if not all, ambiguities have been cleared and that one is more likely to have found a correct interpretation of the sentence, in terms of its syntactic structure, than if the result of the parse consisted in partial analyses only. A diagnosis technique which does not provide a complete parse is not in as good a position to appreciate the quality of its own diagnosis.

Another advantage of constraint relaxation is that no new set of rules needs to be added to the grammar. Adding new rules to a grammar is always problematic for several reasons. (i) For error diagnosis, new rules to be added are often mal-rules. This is highly displeasing on a linguistic standpoint because a grammar should only contain the means to construct grammatical sentences. Mal-rules allow the creation of ungrammatical structures and thus should not be part of a grammar. (ii) Another problem resides in adding these new rules into the rule set without upsetting the balance between the existing rules. Rules are often tried by the parsing algorithm in a very specific order and the new rules must be added at their proper place regarding this order. They must be activated only after the corresponding standard rules are rejected, but not necessarily after all the existing rule set. This interweaving might prove a difficult task. Incorrect ordering of the rules might change the analysis of the sentence and increase the number of overflagging occurrences. (iii) Moreover, increasing the size of the rule set often increases parsing time, as more rules need to be tried out at every stage, and thus diminishes the efficiency of a system.

3.1.9 Problematic issues

Over-generation

Although constraint relaxation can be used with deterministic parsers (Charniak 1983, Chen and Xu 1990), it is most often seen with non deterministic ones. When used with a non deterministic parser, the most problematic issue of constraint relaxation is the increase of the search space generated by constraint relaxation itself. Constraints are normally used to separate what is grammatical and admissible from what is not. The constraints restrict the number of paths to be followed by the analysis process. By relaxing constraints, one opens new paths that need to be explored, which in turn might call on constraints which are possibly relaxed. "Furthermore, syntactic constraints, as well as morphological, semantic, and pragmatic constraints, are sometimes crucial to determining what interpretation is intended, so that ignoring syntactic constraints can lead to seeing ambiguity where there is none" (Weischedel and Ramshaw 1987, p. 159). Depending on the number of constraints that are relaxed, one can easily see how one quickly reaches an explosion of the search space. Thus, while relaxing constraints, one must also think about ways to restrict this over-generation of search paths and, in the end, of structures.

One partial solution, which is already included in the way we have defined constraint relaxation above (section 3.1.5), is to allow a constraint to be relaxed only at times when the unrelaxed constraint would not be satisfied. In this way, the search space is expanded only when the regular path cannot be followed.

One can also limit the number of constraints that are allowed to be relaxed during a single parse and, if needed, parse a sentence several times with different relaxed constraint combinations. This has the advantage of restraining the search space as fewer paths are opened at the same time. However, there are two drawbacks. The first one concerns the error diagnosis process. If fewer constraints are relaxed, fewer errors can be detected in a given pass. One could try to argue that, in this case, one would only need to make several passes with different combinations of relaxed constraints each time. Constraints are, however, linked to one another. There is no guarantee that, even with thoughtful repartition of the constraints into packets, the constraints which are not relaxed would not prevent detection of errors by the constraints which are relaxed. We could be faced, for example, with a word order error and a subject-verb person agreement error in the same sentence (18a). Supposing that the word order constraint is in full force but that the person agreement constraint is relaxed, it is still possible for the system to not detect the error between the subject and the verb. The word order error, for which the constraint is not relaxed, could prevent the parser from reaching a complete analysis of the sentence, which thus returns the chunks shown in (18b), (18c), and (18d); subject and verb could find themselves in two different parts (18b) and (18d), and thus the agreement error between them would not be detected.

(18)a. *The cat well sleep.

b. [$_{\texttt{DP}}$ The cat]

c. [$_{\texttt{AdvP}}$ well]

d. [$_{\texttt{VP}}$ sleep]

A third possibility to contain overgeneration is to limit the size of the search space during parsing by pruning off branches that are not likely to bear any interesting solution. To know which branch to cut, one needs some sort of evaluation method. This is often accomplished via a scoring mechanism. Scores are updated during parsing and each time a path reaches a certain threshold, which can be either a fixed value or a value relative to the score of other branches, that particular branch is cut off and discarded, or pruned, thus limiting the search space. Pruning is often used in search techniques (Sedgewick 1988, p. 627).

Checking three elements

Typically, errors are due to a mismatch of two elements, as in example (16), page 67, or of one element and the rest of the sentence (19), where the error is due to the use of *a* (has) which is homophonous with *à* (about).

(19) *Je pense a Marie.*
I think have Mary.

It sometimes happens that an error occurs because of a mismatch between three elements. This happens in German where noun phrases bear an overt case and some prepositions take different cases depending on the verb they attach to. One must thus know the verb and the preposition to decide the proper case for the noun phrase.

(20)a. *Das Buch liegt auf* ***dem*** *Tisch.*
The book lies on the-dative table.

b. **Das Buch liegt auf* ***den*** *Tisch.*
The book lies on the-accusative table.

The verb *liegen* (lie) takes a prepositional object headed by *auf* (on) which takes a dative complement. In (20a), the sentence is correct, while in (20b) the sentence in ungrammatical because the complement of the preposition is in the accusative case.

On a pedagogical standpoint, how should an error such as (20b) be diagnosed? Is it an error between the verb and the preposition or between the preposition and the noun phrase? Should the three elements be marked down? Apart from the pedagogical ideal, how does a diagnosis system working with constraint relaxation diagnose this type of mistake?

The answer to this question depends partly on parsing strategy. The results are different if the parser is able to attach only complete elements to the structure it is building, or if it can attach partial elements and complete them later with the rest of the input.

(21)a. verb + (auf + DP)

b. (verb + auf) + DP

In the first case, illustrated by (21a), the whole prepositional phrase is computed before attachment to the verb. When the accusative DP *den Tisch* (the table) attaches to the preposition *auf* (on), no error is detected as the preposition takes either the accusative or the dative case depending on its context and the context is unknown at the time. Moreover, it is most likely that the whole preposition phrase takes the accusative case by unification/assimilation with the accusative DP. Thus, when the accusative PP tries to attach to the verb, the case constraint is not satisfied because the verb requires a dative PP. The constraint is then relaxed, the verb and the PP attach to one another, and a case agreement error is marked down between the verb and the PP. If the location of errors is actually not written in terms of projections, but in terms of heads, then the error appears between the verb and the preposition itself, although the verb and the preposition match perfectly by themselves.

In the second case, shown in (21b), the parsing algorithm attaches the preposition to the verb and then completes the preposition phrase with the DP. Attachment of the preposition to the verb does not raise any difficulty as the verb *liegen* requires a dative preposition and as the preposition *auf* bears both the accusative and the dative cases. Through the result of assimilation/unification, the preposition retains only the dative case. When the first element of the future determiner phrase (most often the determiner itself) tries to attach to the preposition, the case agreement constraint is activated. If, as in (20b), the determiner bears the accusative case, the constraint cannot be satisfied and is then relaxed. A case error is marked down between the preposition and the determiner. However, the error here does not exist without the connection to the verb as the structure 'auf + accusative DP' exists in German.

Thus, in the first case, we are marking the verb and the preposition, and in the second the preposition and the determiner. We are never marking the three elements because constraints are verified when one element attaches to another, that is between two elements and not three. If we must mark the error on two elements only, it would seem more interesting to mark the case error between the verb and the DP, which is not one of the options proposed by our two algorithms. Thus, some kind of gymnastics has to be performed in the student module for the final output to indicate that the error is located between the verb and the DP, or to show a link between three elements.

3.1.10 Variants

There are several variants to constraint relaxation as it has been described above. We review three of those variants below. The variants are all based on the same constraint relaxation principles, but the mechanisms allowing constraints to be relaxed are somewhat different in all three cases.

Continuum

The delimitation between hard, soft, and conditional constraints is somewhat arbitrary. There is in fact a continuum in the way these constraints interact with grammaticality. Thus, instead of using such terminology and preventing hard constraints to be violated, one can conceive of a more subtle gradation of constraints. Constraints would then be associated with a weight and the structure resulting from a parse would be assigned a score depending on the combined weights of all the constraints which were violated to build the structure.

This mechanism has the effect of ranking resulting analyses according to a combination of the number of violated constraints and on their relative position on the constraint continuum. The sentence structure with the lowest score is selected as the best choice because it either violates fewer constraints or less important ones. Any constraint can thus be violated, but if its weight is high, the resulting structure is not likely to be selected among the competing alternatives. The difficulty of this variant lies in assigning the weights to the constraints and in designing a scoring algorithm in a way which reflects the link between constraints and grammaticality.

At least two systems using a similar scoring mechanism exist. The first one is the deterministic parser Paragram (Charniak 1983). Paragram employs two weights per constraints: one in case of constraint satisfaction and the other in case of constraint violation. A rule is selected for application if it obtains the highest score computed from all the constraints attached to that rule. The second system is described by Heinecke, Kunze, Menzel, and Schröder (1998), Menzel and Schröder (1998a), and Menzel and Schröder (1998b) as using "graded, hence defeasible, constraints" (Menzel and Schröder 1998b, p. 45). The computing algorithm making use of multiplication and the highest score being the best, a score of 0 indicates a hard, unviolable constraint, a score close to 0 a well-formedness condition, and a score close to 1 is "used for conditions that are merely preferences rather than error conditions" (Menzel and Schröder 1998a, p. 489).

Enforced constraints

Instead of relaxing constraints, one can take the opposite view and enforce constraints. The pedagogical idea is to enforce only the constraints linked to the grammatical phenomena the learner is currently in the process of mastering. Thus, if the focus of attention is on word order, agreement constraints need not be enforced, and errors of this type are not flagged to the users.

This variant of the constraint relaxation technique, implemented by Weinberg, Garman, Martin, and Merlo (1995), requires a very good definition of the hard constraints which are never relaxed, in opposition to the other constraints, which annotate the syntactic structure with feature information and which are enforced only on demand. The parser must be able to provide a sentence structure for an input sentence when none of the constraints are enforced. Relaxed constraints must therefore be limited to constraints which do not change the shape of the syntactic structure. Default values are used when constraints are not satisfied, in order for the parse to proceed smoothly.

Such systems work the other way around from most common constraint relaxation based diagnosis systems. Instead of starting with a constrained parser, from which one relaxes constraints to achieve a complete parse, they start with a completely relaxed parser whose constraints are enforced only on demand. It is more a question of philosophy than of actual performance of the two types of systems, as both provide a certain (flexible) range of relaxed constraints. However, with enforcement on demand, the system's normal working mode is when no constraint is enforced, in opposition to other types of systems, and it is in this mode that one should test it for efficiency. One is perhaps stricter in the evaluation of an 'enforced on demand' parser because it is tested in its worse configuration.

Single and multiple pass diagnosis systems

Diagnosis systems based on constraint relaxation vary also with regards to the number of times a given sentence is passed on to the parser. First of all, there are systems which parse the sentence once with all the constraints enforced to check its grammaticality. If the parse results in a complete analysis, the sentence is deemed grammatical and no further processing is needed by the diagnosis system. This screening is useful to improve the efficiency of the diagnosis system. Parsing a sentence with relaxed constraints is always longer than when all the constraints are enforced. Thus, this first pass filters out all the sentences which do not need diagnosis so that constraints can be relaxed only when needed.

Leaving aside this screening pass, systems use one or several passes through the parser to diagnose errors by constraint relaxation. Single pass systems must have all the constraints linked to errors they are supposed to diagnose relaxed at the same time. Only in this way can the system diagnose all the errors within a single pass. A single pass is usually quicker than multiple passes with fewer constraints relaxed (Vandeventer 2001, p. 118). However, the tradeoff is found in the number of analyses provided by this single pass. Extracting the correct interpretation of the erroneous sentence from several analyses might prove difficult and heuristics not always reliable.

The alternative to the single pass approach is naturally multiple passes. One could contemplate parsing the sentence once for each error type one wants to diagnose, that is for each relaxed constraint. This may not be an optimal solution as seen in section 3.1.9. Relaxing constraints by packets is already more viable as one can organize those packets so that constraints linked to one another are relaxed together. Thus, all the agreement constraints (gender, number, person, case) would belong to the same packet. Moreover, some constraints could be part of several packets if their relaxation proves useful to build more complete structures (and thus diagnose more errors) and not too damaging in the number of spurious structures created. This might be the most effective compromise between single and multiple pass systems.

3.1.11 Remarks

Constraint relaxation is a well-known technique for error diagnosis and it has been used in many prototype systems. Its main drawback, the proliferation of unwanted structures, is an inevitable by-product of the technique. Only careful selection of relaxed constraints and limitation in the number of constraints relaxed at the same time can provide some relief. However, the advantages, in terms of complete parse and facility of extracting error diagnoses, are not negligible. Thus, we consider constraint relaxation a very interesting technique for syntactic error diagnosis, as long as the range of errors it must treat is appropriate in kind and size.

3.2 Phonological reinterpretation

Language learners tend to confuse words which sound alike but have different spellings. Although this type of confusion happens with native speakers as well, it is much more frequent with language learners whose mastery of spelling in their L2 is not as good. There are two phenomena, although interrelated, that we need to distinguish. (i) Phonetic spelling can be defined as stringing together letters which form the sounds of the word one wants to write. Phonetic spelling is used by people who know the phonetic value of letters, or combination of letters, in the language they are writing in but do not know its spelling rules well enough. They are proficient in phoneme-to-grapheme transcription. Words resulting from phonetic spelling can be non words (22a) or they can be legal words (22b), that is, words belonging to the lexicon of the particular language. (ii) The second phenomenon occurs when someone mixes up two words which sound alike and writes one for the other. The word which is written instead of the appropriate one is by necessity a legal word (22c).[3] Moreover, (22c) can also results from phonetic spelling. This error is thus due either to performance or to competence issues.

(22)a. *The dog **wich** I saw was sick.
The dog which I saw was sick.

b. *The tap had a **leek**.
The tap had a leak.

c. *The dog **witch** I saw was sick.
The dog which I saw was sick.

(22a) is an error of a kind usually treated by any spell checker. Unknown words are easily picked up, although interesting correction proposals are

[3]In some dialects of English, 'which' is pronounced /hwɪʧ/ and is not homophonous with 'witch', pronounced /wɪʧ/.

not always provided. (22b) and (22c) result in the same kind of difficulty, irrespective of the reasons or the manners in which the words were produced. In both cases, we are faced with a legal word which has no reason to be in the position it occupies.

The problem with legal words can be further subdivided into two cases. (i) In (22b), the word is of the same lexical category as the one it replaces. "Unfortunately, a parser cannot detect substitutions by homophones which have the same syntactic properties" (Vosse 1992, p. 112). Without a semantic component, it is impossible for a syntactic parser to distinguish that there was a confusion. (ii) In (22c), the two words are of different syntactic categories, which most often prevents the parser from assigning a complete analysis to the input sentence. It is errors of this latter case that we intend to diagnose with phonological reinterpretation. Phonological reinterpretation is not limited to the word level, however. Phonetic spelling can span several words and the technique can be theoretically applied to whole sentences, as we will see.

This section is organized as follows. A definition of phonological reinterpretation is proposed in section 3.2.1. In section 3.2.2, the error types which can be treated by phonological reinterpretation are exposed. In section 3.2.3, prerequisites for phonological reinterpretation are stated. The diagnosis algorithm comes next in section 3.2.4. Advantages (3.2.5) and problematic issues (3.2.6) are then discussed. A variant to the main technique is proposed in section 3.2.7, related systems are briefly described in section 3.2.8. Some remarks (3.2.9) then conclude the section.

3.2.1 Definition

Phonological reinterpretation is based on the principle that sequences that sound alike do not necessarily share the same spelling, and that a misspelled sequence might sound identical to its correct written form. The sequence in question can range form a whole sentence to a single word.

There are two main steps in phonological reinterpretation. (i) The sequence is first transformed into its phonological counterpart. (ii) This phonological sequence is then reinterpreted into a written text. Phonological reinterpretation is defined by the juxtaposition of these two steps.

Phonological reinterpretation can be activated on whole sentences, where it implies heavy mechanisms but should provide exhaustive results, or on very short sequences, with the word as the smallest unit, which require much simpler tools but obviously cannot tackle as many error types.

3.2.2 Errors treated

At the word level, errors which can be treated by phonological reinterpretation include homophones of distinct lexical categories as in (22c). A homophone is "a word having the same sound as another but of different meaning or origin" (Allen 1990, p. 565).

Phonological reinterpretation can also be used for some agreement errors in languages in which agreement is not always overtly realized, such as French. This is a particular application of the technique which is very much language dependent. Indeed, in standard French, many agreement markers, such as feminine *-e* or plural *-s*, are not phonetically realized and the mark can be seen only in writing. However, missing such a marker in a written text is considered to be an agreement errors which requires correction.

(23)a. **Les* ***chat*** *dorment.*
The-plural cat sleep.

b. *Les chats dorment.*
The-plural cats sleep.

c. /le ʃa dɔʀm/

(23a) and (23b) are both pronounced as in (23c).[4] There is no difference at all. By reinterpreting the sentence in (23a), the plural *chats* (cats) can be found.

A multi-word use of phonological reinterpretation concerns sequences of words which should be written only as one (25), or words which should be rewritten as several (26).[5]

(25)a. * an other
another

b. * more over
moreover

(26) * Its a problem.
It's a problem.

At the sentence level, depending on the sensitivity of the parser and of the reinterpretation mechanism, punctuation errors, as in (27), might be diagnosed.

(27) * There were apples, pears, bananas_ and oranges in the fruit salad.
There were apples, pears, bananas**,** and oranges in the fruit salad.

[4]Some of the phonetic transcriptions were composed with the help of the phonetic dictionary compiled by Boë and Tubach (1992).

[5]If the agglutinated word is not part of the lexicon, it should be detected by the spell checker.

(24) * theproblem

3.2.3 Prerequisites

The main idea of phonological reinterpretation is to find alternatives to the written input which are pronounced in the same manner. If the parser is unable to analyze some sentence, we try to figure out if some words were spelled in a homophonous way and should be replaced by another sequence of characters. If the new parse succeeds with the reinterpreted words, it gives us interesting material for error diagnosis.

This technique can be implemented only if some prerequisites on the parser and the lexicon are respected, and/or if some specific tools are available. We now discuss these prerequisites.

Parser

The parser should be able to serve as recognizer. That is, it should fail on all ungrammatical sentences and provide a complete analysis for all grammatical sentences. Only in this way can we know that there was an error in a sentence. When the parse fails because the sentence is ungrammatical, partial analyses of the sentence must still be provided, as it is from these partial analyses that words are selected for phonological reinterpretation.

As several alternatives can possibly result from the reinterpretation of a single word, it is preferable for the parser to be non deterministic. One can thus pass on to the parser all the alternatives and let it sort those which are viable, that is, those which result in a (more) complete analysis, and the others which must be discarded. In case the parser is deterministic, then the sentence must be explicitly fed to the parser with each different combination of word reinterpretation alternatives. This is of course possible, but much work risks being repeated an unnecessary number of times.

Lexicon

The lexicon which is used by the parser must have two specific features which are not required for normal parsing of written input. The first feature is to store the phonological form of each word in the lexicon. This is the quickest way for the diagnosis system to obtain the phonological representation of a given word. The second feature is that the lexicon must be searchable through the phonological information, in order to enable the system to retrieve homophonous forms.

Phonological reinterpretation at the word level implies retrieving the phonological form of a word in the lexicon and searching this same lexicon through the phonological forms. Words with an identical phonological form are retrieved for reinterpretation.

Synthesizer and phonological recognizer

If a whole sentence must be reinterpreted, the system needs more powerful tools than phonological entries in a lexicon. The system needs to be able to provide the phonological string representing the whole sentence. This string must contain not only the words themselves, but also all the information linking the words together, such as liaison and elision phenomena. This implies the use of the component of a speech synthesizer which transforms a string of characters into a phonological representation, a ‘sentence phonetizer’.

The reinterpretation step of the process requires another tool, namely the component of a speech recognition system which would transform a sentence represented phonologically into a string of characters. It would be the reverse operation. Although to go from the input signal to its phonological representation, before segmenting it into words and giving each word its orthographical representation, seems a logical path for a speech recognition systems, most speech recognition systems do not in fact use a phonological representation of a sentence. They propose words directly from the input.[6] Let us however assume, for the sake of discussion, that a tool capable of transforming a phonological representation of a sentence into a segmented orthographical representation exists or could be created.

Whole sentence phonological reinterpretation implies many elements of a text-to-speech-to-text system. These are complex components which are not easily found or created. Moreover, they are resource greedy. Thus, phonological reinterpretation of whole sentences must be employed only when absolutely required in order not to bear too heavy a load on the system resources or to slow down the system noticeably.

3.2.4 Algorithm

One can consider that there is heavy and light phonological reinterpretation. Heavy reinterpretation involves whole sentences while light reinterpretation is concerned only with unconnected words. There is in fact a continuum between those two extremes, ranging from a whole sentence, to a few words, and to a single one. If several unconnected words belong to the same sentence, they are reinterpreted at the same time.

A sentence is first parsed by the regular version of the parser. If the sentence is analyzed completely, it is deemed to be error-free. If only partial analyses result from the parse, a choice must be made between heavy and light reinterpretation. The algorithm for heavy phonological reinterpretation

[6] “Either whole segments are directly recognised [by speech recognition systems](global method), or an intermediate phonetic labelling is used before lexical search (analytic method. ... Most commercial methods implement global techniques” (Chollet 1994, p. 140).

is given below.

1. The whole sentence is phonetized. This requires the use of the component of a speech synthesizer specialized in transforming a character string into its phonological representation.[7]

2. The whole sentence is reinterpreted as if it were a phonological transcription of speech. For this, one must make use of a component of a speech recognizer able to transform a phonological representation into a string of characters with word separations and punctuation marks.

3. Finally, one must compare the input string with the output string and check them for differences. Places of modifications are the indication of errors in the input string and a diagnosis can be formulated on their basis.

Heavy phonological reinterpretation is difficult, not to say impossible, to obtain because of the tools it requires. Moreover, the process of speech recognition is full of ambiguity. Heavy reinterpretation is thus very likely to prove cumbersome and lacking in efficiency.

When heavy phonological reinterpretation is not feasible, because components are not available or resources are too scarce, lighter phonological reinterpretation can still be performed. The algorithm for light phonological reinterpretation is given below.

1. Words located at the borders of partial analyses (apart from the first and the last words of the sentence, unless they form a partial analysis of their own) are submitted for word phonological reinterpretation. For each word:

   (a) the phonological string of the word is looked up in the lexicon and retrieved;

   (b) the phonological string is used to search the lexicon, through the phonological entries, for identical words;

   (c) phonologically identical words with the input word, or homophones, are retrieved with all their lexical information;

2. If no homophone was found, then no diagnosis is provided and the parse fails;

3. If one or several homophones are found, then they are added as alternatives in the non deterministic parser, with an error mark, and the sentence is parsed again;

[7]Describing the components of speech synthesizer or recognizer is beyond the scope of this dissertation.

4. If the parser is now able to provide one (or several) complete analyses of the sentence, the homophone(s) retrieved from the lexicon and used in the complete analysis are set apart as correction proposals.

5. Correction proposals are compared to their corresponding input words through their lexical information. Error diagnoses are extracted from this comparison.

3.2.5 Advantages

As with constraint relaxation, phonological reinterpretation provides the users with a complete analysis of the input.[8] The advantages implied by complete analysis in the CALL context are outlined in section 3.1.8. There is, however, a slight difference. While the input is not modified with constraint relaxation, phonological reinterpretation makes use of words which were not part of the original input and are substituted to the erroneous words. Thus, correction, or at least correction proposals, is an inherent part of phonological reinterpretation. There is no phonological reinterpretation if a word or series of words cannot be proposed for substitution in the problematic area. The reinterpreted words, if they allow the sentence to be completely analyzed, are the correction proposals to be submitted to the users.

Phonological reinterpretation can also be used in spell checker applications. The checking part itself is uninteresting as it is sufficient to compare each input word to those stored in a lexicon. Suggesting possible corrections for unknown words requires more skills. Phonological reinterpretation can be applied to single words in the process of finding correction proposals. In this case, as the starting word is not part of the lexicon, by definition, a phonetizer must be used to figure out the phonological form of the input word. A phonetizer can be viewed as an expert system which contains the grapheme-to-phoneme transcription rules of the language in use and can thus assign a phonological string to an input word (Gaudinat and Goldman 1998, p. 140). Once the phonological transcription is achieved, the process is the same as the one described in the light reinterpretation algorithm: there is a dictionary look up through the phonetic entries and the corresponding homophonous words are retrieved. Thus, an exact correction proposal can be given for an unknown word which was spelled in a homophonous way to the word desired. Moreover, if the dictionary lookup is capable of splitting its input in several words, this technique also allows for good recuperation of agglutinated words (see example (24) in footnote 5) which often receive correction suggestions which have not much in common with the desired string of letters in common spell checkers.

[8]The original input word must simply be replaced in the sentence analyzed with the reinterpreted word.

3.2.6 Problematic issues

Latent consonants

One of the interesting problems linked to phonological reinterpretation is the influence of words on one another within an utterance. Words do not necessarily have the same phonological transcription taken in isolation or in the context of a sentence. This is due in good part to liaison and elision phenomena, and to latent consonants. Thus, language learners, for whom word boundaries might not be as evident as it is for native speakers, might spell a word as it is pronounced in context rather than in isolation. They might also omit the last consonant of a word if it is latent. The dictionary lookup procedure which retrieves words by their phonetic entries must thus be able to account for these phenomena.

(28)a. ****Cet*** *un compromis.*
This a compromise

b. *C'est un compromis.*
This is a compromise

The erroneous word in (28a), corrected in (28b), finds its phonetic transcription in (29a). It can obviously be rewritten as the same word again. But it can also be reinterpreted as other words, as demonstrated by examples (29b) and (29c). Moreover, *cet* ending with a consonant, it is possible that the word we are looking for ends with the same consonant, not pronounced however when the word is taken outside of context. Thus, the standard phonetic transcription of the corresponding word will not contain the final consonant. It must however be entered in the lexicon as a latent consonant used for liaison purposes (30a). Possible reinterpretations of this kind are shown in examples (30b) and (30c).

(29)a. /sɛt/

b. *cette*
this-feminine

c. *sept*
seven

(30)a. /sɛ+t/

b. *sait*
knows

c. *C'est*
It is

Multi-words

There is also the question of run-on-words, or agglutination. In this case, a single word must be transformed into several. This is also exemplified in (28) where *cet* (this) is corrected by two words. The algorithm must thus be able to find all the words and word combinations which correspond to a given phonetic string.[9]

The reverse is also true. Sometimes a word is spelled in two words while it should be only one.

(31)a. **Les avantages sont* ***plus tôt*** *psychologiques.*
The advantages are more early psychological.

b. *Les avantages sont plutôt psychologiques.*
The advantages are rather psychological.

Being able to regroup words implies the application of the phonological reinterpretation techniques to several words together, and most probably in regrouping those words across partial analyses boundaries. Moreover, if latent consonants exist as part of the words to reinterpret, the algorithm must try liaison phenomena in addition to regrouping two words into one. Thus, for the erroneous sequence *plus tôt* in (31a), one must look up for words with the phonetic transcriptions in (32a) to (32h).

(32)a. /ply/ + /to/

b. /ply/ + /tot/

c. /plyto/

d. /plytot/

e. /plys/ + /to/

f. /plys/ + /tot/

g. /plysto/

h. /plystot/

[9]Given that these compounded multi-words must be legal words (otherwise they would have been detected by the spell checker), their actual number might be quite small. If this proved to be the case, one could imagine setting up a list of those words and their correspondence. Searching such a list might be more efficient that multi-word reinterpretation.

With only two short phonetic sequences, which can form one or two words, both sequences having a latent consonant, the number of possible combinations of words to be retrieved reaches 2^3, that is 8 possibilities. Of these, only some will find their equivalent in the lexicon. Moreover, some phonetic sequences, such as (33a) can be transcribed into several possible words (33b) and (33c):

(33)a. /to/

b. *tôt*
early

c. *taux*
rate

We can easily see that the level of ambiguity is high and that one should limit the number of words that need to be reinterpreted together to prevent a combinatorial explosion. In this example, only (32c) is the correct version which can be retranscribed into *plutôt* (rather).

High frequency errors

In French, there are also two pairs of especially problematic homophonous words: *a*/*à* (has/to) and *ou*/*où* (or/where). Only an accent, not phoneticaly realized, separates these couples of words which have different lexical categories and meanings. Accents are often forgotten by language learners and, in a lesser way, by native speakers as well. Moreover, because one is drilled to think of these words as problematic, overcorrection sometimes occurs, and accents are also present when they should not be. Thus, the frequency in which these words are used erroneously is high. Mistakes with these words can easily be caught by phonological reinterpretation. However, the expense of phonological reinterpretation might be too high for such a known and frequent mistake. Thus, for efficiency reasons, it might be more interesting to treat these pairs of words in an *ad hoc* manner rather than to use the technique specially designed to treat this kind of words. This applies naturally to French for these two words, but similar groups of words can certainly be found in most languages, like in example (34) for English.

(34) to — two — too
/tuː/

3.2.7 Variant

A possible variant to phonological reinterpretation as described so far is to expand the search for homophones in the lexicon to near homophones or close enough neighbors. By that, we understand words that sound much, but not quite, like the original word. One must therefore introduce a distance measure between the phonological transcriptions of the words. Instead of retrieving from the lexicon only the words with a distance of 0 (true homophones), one can retrieve all the words up to a certain threshold. This enables close sounding words to be taken into account as well.

Obviously, using a distance metric allows many more words to be retrieved from the lexicon and thus increases the ambiguity level, giving rise to more possible alternatives. Therefore, one might not want to be as permissive as that. A rough metric which computes the distance by counting the number of changed phonemes might still be too strict, even if the allowed distance is only of 1. There might be better, language dependent, heuristics. In French, we might be able to isolate specific difficulties encountered by many language learners. For example, the difference between nasal vowels is perceived with difficulty by many learners of French, which might explain the authentic error in (35a), corrected in (35b).

(35)a. **Une* ***langue*** *histoire*
/yn lɑ̃g istwaʀ/
A tongue story

b. *Une longue histoire*
/yn lɔ̃g istwaʀ/
A long story

On the basis of example (35), one could consider all nasal vowels to have the same value for phonological reinterpretation, or only /ɑ̃/ and /ɔ̃/, as well as /ĩ/ and /ɛ̃/, to be interchangeable. Thus, we would not be using a distance metric of 1, but tailor the extension of what is acceptable as homophonous to specific cases we know raise problems with language learners. Similarly, there are many ways to write the phonemes /e/ and /ɛ/. They sound quite close to one another to foreign ears, although they are minimal pairs and differentiated by native speakers, depending on the specific dialect they speak.

(36)a. **La langue* ***et*** *le moyen unificateur.*
/la lɑ̃g e lœ mwajɛ̃ ynifikatœʀ/
The language and the means unifying.

b. *La langue est le moyen unificateur.*
/la lɑ̃g ɛ lœ mwajɛ̃ ynifikatœʀ/
The language is the means unifying.

Both examples (35a) and (36a) have only minor phonological changes compared to the correct sentences (35b) and (36b), respectively. The distance for the looked for word is only of 1, as only one phoneme was changed.

Thus, the correct word could be reinterpreted if we used a rough distance of 1. However, we would also get many unwanted words, and it is probably much better to limit the use of approximation for phonological reinterpretation to a few specific cases, deciding which ones to treat for a particular language based on corpora studies. For French, they should most probably include differences in nasal vowels, in the openness of vowels, and of their length.

3.2.8 Systems using elements of phonological reinterpretation

Elements of phonological reinterpretation have been expressed before and used in spell and grammar checkers. However, to our knowledge, they have not been gathered, or described, into a unified technique before.

Courtin, Dujardin, Kowarski, Genthial, and de Lima (1991) use phonetics at the lexical level. For unknown words, they compute "all the possible phonetic transcriptions associated with the string" (Courtin et al. 1991, p. 159). From these, they generate all the corresponding written forms. They filter the list by checking it with a lexicon and in retaining only legal words. Thus, their system uses phonetics to retrieve correctly spelled words. However, it does not make use of the phonetic information which can be stored into a dictionary entry. They rely only on a complete grapheme-to-phoneme bidirectional correspondence.

Vosse (1992), in his spell and morpho-syntactic corrector for Dutch, proposes to parse sentences with all possible correct entries for a misspelled word. Only the entries resulting in complete analyses are retained as possible corrections in the sentence context. "In order to handle errors caused by homophones as well, this mechanism needs to be extended. When dealing with legal words it should use their syntactic categories plus the syntactic categories of all possible homophones, plus — to be on the safe side — every alternative suggested by the spelling corrector" (Vosse 1992, p. 114).[10] While this might be rather extreme given the number of alternative paths that will crop up during parsing, it nonetheless brings to the forefront two very important concepts. (i) One can retrieve the information contained in words which are homophones to those of the sentence, and parse the sentence with this new lexical information as well. (ii) The parser can act as filter to the proposed correction alternatives.

The Microsoft® Word 97 English grammar checker uses these two concepts as well. This system does not use phonology directly, but a list of pairs of words which are frequently taken one for the other has been built up. "When a word that is in this file appears in text, the analysis system

[10]Unfortunately, Vosse (1992) does not explain in the cited article how his system is able to retrieve homophones and their lexical categories.

behaves as if the other word of the pair also appears at that same position in the text. Because of the bottom-up, multipath nature of the parsing algorithm, typically, just one or the other word of the pair will result in a parse. If the word that results in a parse is not the word in the text, but rather the other one, then that other word is suggested to the user as a substitute for the word that was typed" (Heidorn 2000, p. 193). All the post-processing devices needed after phonological reinterpretation are used in this system. It is only lacking phonological reinterpretation itself, which would make it more general.

Courtin et al. (1991) and Vosse (1992) thus laid the foundations for phonological reinterpretation about ten years ago. They do not seem to have had many followers, apart from (Heidorn 2000) nearly a decade later who made use of the idea of parsing with substitute words and filtering the alternatives with parsing. When phonology is used, it seems to be only for the lexical level of checking. Using homophones or or a list of near words as alternatives during parsing does not seem to be linked to phonological reinterpretation at the syntactic level. Moreover, we do not know of other literature which would indicate further progress towards a more complete system working along the principles of phonological reinterpretation as we described it in this section.

3.2.9 Remarks

Heavy phonological reinterpretation will prove most certainly too difficult to implement, given the complex components it uses. Its implementation and use is probably not worth the effort and the resources that it would require, when considering the range of error types which would be diagnosed.

Although phonological reinterpretation covers only a relatively small range of error types, its use is interesting in the CALL context because it is able to diagnose error types made by language learners and which other techniques, often more widely used, are not diagnosing properly.

Because phonological reinterpretation can, by definition, only diagnose errors which do not produce a sound change (with the exception of the variant described in 3.2.7), it is incapable of diagnosing many other error types that language learners are likely to commit. Thus, phonological reinterpretation is better used in conjunction with other diagnosis techniques. It can therefore specialize in what it is good for, that is, homophones and agglutination phenomena, and let techniques better suited for other error types deal with them.

3.3 Chunk Reinterpretation

When a parser is not able to reach a full analysis for a given sentence, it can provide, depending on its parsing algorithm, partial analyses or chunks.

Put side by side, these chunks cover the whole sentence. The basic idea underlying chunk reinterpretation is to examine chunks in order to discover why they could not combine into a full analysis. We assume that the reason is directly linked to grammar errors.

This section is organized as follows. We start by defining chunks in section 3.3.1. In section 3.3.2, we lay the foundations of chunk reinterpretation. We state the error types which can be diagnosed with this technique in section 3.3.3. Section 3.3.4 exposes the prerequisites for the technique and section 3.3.5 the algorithm. We then propose a schematic rule set in section 3.3.6. Advantages (3.3.7) and problematic issues (3.3.8) of chunk reinterpretation are then discussed. We continue by presenting a possible variant to the technique in section 3.3.9, and conclude with some remarks in section 3.3.10.

3.3.1 Definition of 'chunk'

'Chunks' and 'partial analyses' are synonym words and are used interchangeably in this section. Although we can have a rather good intuition of what they are, we must define them more formally.

One of the goals of syntactic parsing is to provide a syntactic analysis of the input sentence. It is sometimes not possible to compute an analysis that covers the whole sentence, either because the sentence is outside the bounds of the parser or because it is ungrammatical. However, much analysis work has been done over the sentence and pieces of it have been analyzed. Instead of returning simply that the sentence has not been fully analyzed, a parser can return the results of its work so far, that is, chunks of syntactic analysis or partial analyses which do not cover the whole sentence.[11]

Through non deterministic parsing,[12] many partial analyses are created. Not all are presented as the result of the unsuccessful parsing process. Chunks are selected through several criteria. However, the only compulsory one is that the chunks, put one after the other, must cover the whole sentence. Other criteria are left to the parser's implementation and are not discussed here.

The size of these chunks may vary greatly. The smallest chunks must cover at least one word. The longest can cover up to the whole sentence, minus one word. Indeed, if the chunk covered the sentence entirely, it would not be a chunk any more, but a full analysis. In the worst situation, the parsing process provides as many chunks as there are words in the sentence. In the best situation, only two chunks cover the whole sentence.

[11]A parse which does not lead to a full analysis creates many chunks of different sizes. For efficiency reasons, they cannot all be examined. A selection of the best series of chunk covering the whole sentence is made based on the chunks' length.

[12]Chunks may be provided by a deterministic parser as well.

3.3.2 Chunk reinterpretation

The principle behind chunk reinterpretation is that there must be a reason why two chunks did not combine into a larger analysis. Assuming that the parser coverage is large enough, the reason must be linked to some ungrammaticality of the sentence. Break points between chunks are possible error locations. By examining the chunks, two by two, one should be able to provide a diagnosis for the detected errors.

One must be aware that not all chunk boundaries reveal syntactic errors. Depending on the location and words bearing the error, a single error can result in more than two chunks. In example (37), the number error between the determiner and the noun prevents a complete analysis of the subject of the sentence which therefore cannot attach to the tense phrase (TP). This sentence composed of three words is thus analyzed as three chunks (37b), while there is only one error in the sentence.

(37)a. **A cats slept.*

b. $[_{\mathtt{DP}}$ a $]$
$[_{\mathtt{NP}}$ cats $]$
$[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ ]\ [_{\overline{\mathtt{T}}}$ slept $[_{\mathtt{VP}}\ ]\ ]\ ]$

While there are some similarities between chunk reinterpretation and parse fitting as described by Jensen, Heidorn, Miller, and Ravin (1983),[13] the two techniques are quite different. Instead of observing the chunks as we propose to do in order to determine why they did not combine, Jensen et al. (1983) 'fit' chunks into their most likely position into the parsed structure in order to obtain a complete, if not fully grammatical, structure. Once all the parts have been fitted, error detection rules are invoked to track syntactic and stylistic errors.

3.3.3 Errors to be treated

Errors that can be treated by chunk reinterpretation are, *a priori*, all errors which result in partial analyses or chunks. However, chunks by themselves only detect potential error locations. They are not clear evidence of an error, and they do not indicate the type of the error we are dealing with. This is achieved through chunk reinterpretation rules. Thus, only error categories for which rules have been devised receive a diagnosis. Moreover, chunk reinterpretation tends to work better on local errors than on distant ones because chunk boundaries are used as error locators.

[13]A shorter version of this article can be found in (Jensen, Heidorn, Miller, and Ravin 1993).

There are, nevertheless, some error categories which present themselves as good candidates for chunk reinterpretation. These include agreement in gender, number, and person, as well as verb complementation (38) and 'auxiliary and past participle' erroneous structures (39), as long as these errors provoke partial analyses.

(38)a. *He gave a book **at** him.

b. He gave a book to him.

(39)a. *Il **à** dormi**t** longtemps.
He to slept-preterite a-long-time

b. Il a dormi longtemps.
He has slept a-long-time.

Other error categories which can be treated by chunk reinterpretation comprise errors which produce extra elements in the sentence, especially if this extra element is isolated in a chunk of its own. By 'extra elements', one means phrases which are not essential to the sentence and which are not in their legitimate position. Thus, wrong ordering of adverbs (40a) is an ideal candidate, as well as word doubling (40b).

(40)a. *I **tomorrow** will go.

b. *I have given it **to** to you

(41)a. I will go.

b. I have given it to you.

In each of the sentences in (40), one can omit the extraneous word to render the sentence grammatical, as in (41).

Punctuation errors, the missing comma in (42a), which are frequent in language learner productions, may also cause partial analyses (42b). Partial analyses of specific syntactic categories, juxtaposed side by side, may be an interesting indication.

(42)a. **Il pleut allons au cinéma.*
it is-raining let's-go to-the movie-theater

b. $[_{\mathrm{TP}}$ *Il pleut* $]$
$[_{\mathrm{TP}}$ *allons au cinéma.* $]$

3.3.4 Prerequisites

The essential prerequisite of the chunk reinterpretation technique is a parser which outputs partial analyses when it does not find a complete analysis for a given sentence. As chunk reinterpretation is based only on one set of partial analyses covering the whole sentence, the parser does not need to be non deterministic. In case of non determinism, however, the chunk selection process should hopefully provide the most likely partial analyses, although it might be very difficult to decide which set of partial analyses is more likely than the others.

The parser must also behave as a recognizer in order to produce partial analyses only when a sentence is not grammatical. We have seen in example (38) that a single error can create more than one break in the analysis and, thus, more that two chunks. Therefore, one cannot warrant in any case that a particular break between two partial analyses is directly caused by an error. However, every break between chunks must be treated as a possible error location and chunk reinterpretation must be attempted. Breaks that are not directly caused by an error require extra resources which could be better used elsewhere. Therefore, the fewer breaks which are in fact false alarms, the better for the reinterpretation process. The parser must provide analyses which cover whole sentences and phrases as long as these are grammatical.

Moreover, a set of chunk reinterpretation rules is needed in order to try the different reinterpretation possibilities. This is not really a prerequisite, but rather an integral part of the technique.

3.3.5 Algorithm

When one looks for the reasons why two chunks a and b did not combine into a larger one, there are at least two possibilities which need to be taken into account to locate the attachment point. a can be a specifier of b, or b can be complement or adjunct of one of the elements of a. The first case is relatively simple: the whole of a should have attached in the specifier position of b. In the second case, the whole of b can attach to different non-saturated elements of a, along the right edge of its tree structure. Depending on the length of a, this might mean quite a long structure to examine with many potential attachment points. However, most likely attachment points are the right most node of a, the head of a, and the tensed verb (if there is one) in a.

Error diagnosis is performed on potential attachment points through a set of rules taking into account the attachment point category and features as well as those of the attaching phrase.

Because there can be more than two chunks for a sentence, it is important to repeat the process as many time as there are breaks between chunks in order to diagnose all the errors of the sentence. However, even if some chunks remained isolated, if they cannot be attached to the main structure, and if no error can be found to explain their isolation, it is still possible to provide error diagnosis for the other chunks of the sentence.

Given a list l, containing the chunks (at least two) recuperated from an unsuccessful parse covering the whole sentence, the algorithm for chunk reinterpretation is the following.

```
1. Set the current element of $l$ to the first element of the list.
2. If at least one other element follows the current element,
   (a) retrieve the current element as $a$ and the following one as $b$.
3. Else exit from procedure
4. For each possible attachment point between $a$ and $b$ do
   (a) For each rule of the corresponding set
       i. Activate the rule
       ii. If the attachment is possible
           A. Combine $a$ and $b$ into *new*
           B. Note the error according to the formalism in use
           C. Replace $a$ and $b$ by *new* in the list $l$
           D. Repeat from 1
5. The current element is moved one forward in the list $l$
6. Repeat from 2
```

Diagnosis is complete if the list l contains only one element at the end of the process. If there are more than one element in l, then some errors could not be diagnosed. Partial diagnosis can however be provided to the users.

This algorithm is very dependent on the rule set. It should permit to reconstruct a full sentence structure while diagnosing syntactic errors, as long as the structures of partial analyses do not need to be modified during the combination process.

By reentering the algorithm from the start each time a combination has succeeded and by replacing the elements by their combined version, it should be possible to cover all the chunk boundaries after recombination, thus allowing multiple attachments from right to left. The sentence in example (38) could thus receive a full analysis in the end.

3.3.6 Rule set

The rule set must be ordered to reflect the hierarchical structure of language. When one looks at a chunk, one can consider the phrase most local to the break point, the whole chunk or intermediate phrases. Rules to apply to these different parts are not the same and may contradict themselves. Thus it is important to set an order in which the rules are activated.

To compile the rule set described in this section, we examined an extract of the FRIDA learner corpus[14] parsed with the Fips parser (see section 4.1 for a description of the Fips parser). The rules are therefore designed for French. The schematic rule set can be divided into six groups which should apply in the given order.

- Word doubling (40b)
- Agreement (37)
- Auxiliary and past participle (39a)
- Verb selection (38a)
- Adverb order (40c)
- Punctuation (42)

Word doubling

Rule 1 If the rightmost word of the left chunk and the leftmost word of the right chunk are identical, then issue a word redundancy error message.

Agreement

Rule 2 If the rightmost word of the left chunk is a determiner and the left chunk starts with a noun phrase, and if they do not agree, then issue an agreement error message.

Rule 3 If the left chunk ends with an adjective phrase with the prenominal feature and the right chunk starts with a noun phrase (or if the left chunk ends with a noun phrase and the right chunk starts with an adjective phrase with the postnominal feature) and if they do not agree, then issue an agreement error message.

Rule 4 If there is a noun phrase at the end of the left chunk and the right chunk is a subject-less sentence, and if the noun phrase and the tensed verb do not agree, then issue an agreement error message.

[14]The FRIDA corpus is collected by the Centre for English Corpus Linguistics, *Université Catholique de Louvain*, Belgium. See section 4.2.1 for a description of the FRIDA corpus and its web site (last accessed on April 8, 2003):
http://www.fltr.ucl.ac.be/fltr/germ/etan/cecl/Cecl-Projects/Frida/gateway.htm

Auxiliary and past participle

Rule 5 If the rightmost word of the left chunk is an auxiliary and the head of the right chunk is a past participle, and if the auxiliary is not the right one for the given verb, then issue an auxiliary error message.

Rule 6 If the rightmost word of the left chunk is an auxiliary and the head of the right chunk is a verb not in the past participle, then issue a tense error message.

Rule 7 If the rightmost word of the left chunk is 'à' and the head of the right chunk is a verb not in the infinitive, then issue a homophone error message on 'à' and a tense error message on the verb.

Verb selection

Rule 8 If the left chunk is a subjectless sentence, the right chunk is noun phrase, and the tensed verb in the left chunk and the noun phrase agree, then issue an inversion warning message.

Rule 9 If the left chunk ends with a verb phrase whose verb is not saturated (i.e. it does not have all its complements) and the right chunk starts with a noun phrase or a preposition phrase, then issue a verb complementation error message.

Adverb order

Rule 10 If the left chunk or the right chunk is an adverbial phrase and if there are other chunks on both sides of it, then issue an adverb order error message.

Punctuation

Rule 11 If the left chunk is a noun phrase or a preposition phrase and the right chunk is a sentence with a subject, and if they are not separated by a punctuation mark, then issue a punctuation warning message.

Rule 12 If the left and the right chunks are complete sentences, and they are not separated by a punctuation mark, then issue a punctuation warning message.

The rules sketched here need to be refined depending on the specific parser they are used with. Warnings are used instead of error messages in rules 8, 11, and 12 because subject-verb inversion and punctuation are stylistic issues for which strict rules may not account for all cases and contexts. The size of the rule set should also be expanded in order to include more error types, depending on which error types one wishes to diagnose and, possibly, the other techniques involved in an overall diagnosis system.

3.3.7 Advantages

Chunk reinterpretation has the great advantage of not modifying the parser itself, as it works on its output only. This tremendously simplifies the implementation work as everything which is added can be worked upon separately from the parser itself. For this very reason, the chunk reinterpretation technique is particularly well suited for use with an existing parser that one does not want to, or cannot, modify as one wishes.

Moreover, overgeneration of structures is kept completely at bay as the parser does not provide, and does not try to provide, any more analyses due to the error diagnosis technique. Thus, parsing time itself remains the same and only reinterpretation time, which depends on the number of chunks, needs to be added with chunk reinterpretation.

Finally, the parser does not need to be non deterministic, on the contrary to what is preferred with constraint relaxation and phonological reinterpretation. A deterministic parser would also be able to provide partial analyses for an ungrammatical sentence and it is all that is needed for chunk reinterpretation. Of course, non determinism provides multiple series of partial analyses to choose from and, in the end, a better set of chunks may be provided with a non deterministic parser.

3.3.8 Problematic issues

Chunk reinterpretation is not ideal in many respects. It implies redoing much of the parsing work. There is, first of all, the question of locating possible chunk attachment points, which implies going through pieces of syntactic structures. Once a possible attachment point is located, conditions need to be checked to verify if there is a mismatch of features which prevents attachment in the particular case. These conditions must first have been checked and failed during the parsing process in order for the sentence to have been rejected. If an error is located, then the two chunks are combined in much the same way as the parser would have done it. As we can see, there is much redundancy between work accomplished by the parser and by the chunk reinterpretation technique.

Establishing a good set of chunk reinterpretation rules is far from obvious. The rules themselves are by necessity very dependent on the parser used and on the chunks it provides. It is thus difficult to generalize this set of rules to make it usable with other parsers. Moreover, as with any rule set, it is difficult to ensure that all possibilities have been taken into account and that the errors detected match those described in the specifications.

Furthermore, because one does not need to modify the parser itself with the chunk reinterpretation technique, it is even more important than with the other techniques that the parser acts as a true recognizer. If it accepts ungrammatical sentences and does not provide chunks for them, one is left without the essential raw material on which the whole technique is built.

3.3.9 Variant

A possible variant to chunk reinterpretation as presented in this section so far would be to diagnose errors without necessarily rebuilding a complete sentence structure. This would simplify the technique while still providing error diagnosis. In particular, there might be occasions where chunk reinterpretation would force a modification of the syntactic structure of a chunk. This seems particularly difficult to realize, but nevertheless necessary if one wishes to combine the chunks. Moreover, one would not be able to perform multiple diagnoses from right to left, as proposed in the algorithm (3.3.5), because the chunks would not be modified.

This would raise the problem of not proposing a complete syntactic structure for ungrammatical sentences. We have said before (sections 3.1.8) that full analyses were useful to the users for other NLP tools, such as a sentence structure viewer, which could ideally combine with the output of the diagnosis system to provide information on the sentence structure.

The algorithm proposed above (3.3.5) would have to be modified along the following lines.

1. Set the current element of l to the first element of the list.
2. If there is at least one other element besides the current element,
   (a) retrieve the current element as a and the following as b.
3. Else exit from procedure
4. For each possible attachment point between a and b, activate the corresponding set of rules
5. The current element is moved one forward in the list l
6. Repeat from 2

3.3.10 Remarks

Chunk reinterpretation is a technique very dependent on the parser which provides the chunks. Thus, it is difficult to make generalizations and a particular rule set must be defined for each application of the technique. The rule set must be carefully designed to be precise enough and to cover the relevant error types and chunk configurations with a minimum of rules in order to restrain processing time.

Rebuilding the full syntactic analysis from its chunks is a complex matter as soon as one of the chunk internal structure needs to be modified. This can happen for example with words which have potentially several lexical categories, with verbs whether they are tensed or not, and with some homophones. In all these configurations, chunk re-combination is not advisable as changing one part of a chunk structure may imply changing the whole structure. Therefore, rebuilding the sentence structure, as proposed in the main version, is perhaps not advisable and one should consider using the variant, proposed in section 3.3.9, where the full analysis is not reconstructed.

As it is very difficult to make sure of the coverage of a technique using a set of rules, chunk reinterpretation is better used in combination with other techniques. It is easier to design rules for a specific, restrained set of error types and let the other types, not covered by chunk reinterpretation, be treated by more adequate techniques. As chunk reinterpretation does not necessarily provide a full sentence analysis, it should be used after techniques which do so because, if we reach a full analysis with them, there is no need for chunk reinterpretation and one can use the complete analysis in other applications.

3.4 Scoring

A non deterministic parser provides all possible analyses for a given input. The use of constraint relaxation or of phonological reinterpretation has the effect of increasing the number of possible analyses. "The need for a method of ranking multiple parses in order to select the best one (...) is acutely felt" (Heidorn 1993, p. 49). However, selecting the best analysis is not an easy task. We must first define what we mean by a 'best analysis'. In the CALL context, and as Felshin states it, "the goal of recognition is to produce the most likely interpretation(s) of the input, not the most correct interpretation(s)" (Felshin 1995, p. 263).

One must find the right balance between promoting sentence analyses in which no error has been detected, but which are unlikely given the language learner profile and the complexity of the analysis, and promoting more likely analyses containing some ungrammaticality. It is important to find grammatical analyses for learners' sentences so as not to discourage them by presenting them with error messages when their input is correct. However, learners often produce errors and so one should not bar out all structures containing errors, either. One must thus find a selection metric taking into account the likelihood of a specific sentence analysis, the errors detected in the sentence, and the rarity of its lexical items.

We briefly discuss methods commonly used to assess syntactic likelihood in regular syntactic parse in section 3.4.1. We then propose a method using detected error frequencies in section 3.4.2, and continue with a second proposal employing word frequencies in section 3.4.3. We conclude by explaining how to combine multiple scores in section 3.4.4.

3.4.1 Syntactic likelihood

Different syntactic analyses for a given sentence are not all as likely. Moreover, some structures can be found more frequently in specific text types. Indeed, it seems logical that the kinds of structures to be found in a newspaper or in a children's book are not the same. Likewise, sentence structures produced by language learners do not necessarily correspond to other kinds of data. Depending on their end application, non deterministic parsers, capable of providing several analyses for a single sentence, must often propose only one analysis. Parsers must therefore select the most likely alternative analysis. Mechanisms for ranking alternatives are thus often part of non deterministic parsers. These mechanisms can be based on heuristics coming from long practice or derived from corpus studies, or they can use statistical methods.

The complexity of a sentence structure, from which a score is frequently derived in order to rank alternatives, is composed of the complexity of its subparts and of their number. Thus, each phrase can have its own score, which is combined to the score of the phrase it attaches to. Specific kinds of attachment can be rewarded over others, which would be penalized, in order to promote specific, more likely structures over others.

Statistical methods require very large tagged corpora over which bigram (or n-gram) frequencies are computed and stored for later use. Presented with a sentence, one computes its likelihood by combining the bigram frequencies for every bigram of the sentence. Multiplication is often used as a combination method. The higher score represents the most likely sentence. Results necessarily depend on the adequation of the training corpus to the input sentences.[15]

Ranking alternative structures is a necessity in most non deterministic parsers, independently of its status as an error diagnosis system. Techniques to do so are thus not part of this dissertation. Moreover, ranking procedures are very often designed in complete symbiosis with the parser and nothing needs to be modified for CALL. In fact, the heuristics can be so imbedded in the system that only a full rewriting of the system would permit to modify them in a coherent manner. Obviously, using learner corpora to extract the heuristics or the statistical data capable of selecting more likely structures for the sentences produced by learners would be a plus. There are however other means to do this, as attested by the next two subsections (3.4.2 and 3.4.3), without altering or modifying the already present sentence analysis

[15]More linguistically interesting statistical methods could make use not only of the lexical categories of the words, but also of information on the syntactic structures proposed by the parser. Statistical scoring methods are however outside the bounds of the present work. For a reference on statistical parsing, see (Manning and Schütze 1999) and more particularly chapter 12 (p. 407 ff.) for methods for "probabilities for choosing between parses" (Manning and Schütze 1999, p. 409).

ranking procedures.

3.4.2 Detected error frequencies

While heuristics on the sentence structure and statistical means can be implemented for all kinds of parsers, basing a metric on detected errors can only be part of a diagnosis system. We want to promote grammatical sentences over ungrammatical ones because, if language learners tend to make more mistakes than native speakers, they still do produce many grammatical sentences and it would be a pity not to find those. "The degree of grammaticalness is a measure of the remoteness of an utterance from the generated set of perfectly well-formed sentences" (Chomsky 1964, p. 387). One could consider that each detected error brings a penalty to the sentence analysis. The sum of the errors of an analysis would be its score. The analysis with the fewest errors would be selected. There are three main drawbacks to this view point. (i) Language learners tend to make mistakes and sentence analyses containing some errors may be more likely than error-free structures. (ii) The size of the sentence is a factor that should not be forgotten as the likelihood of errors increases with longer sentences. (iii) Not all error types have the same frequency and this should also be taken into account.

Because it is likely that sentences contain errors in the CALL context, one can wonder whether all errors should be penalized or whether a set number of errors should go unpenalized. However, this would stop promoting completely grammatical sentences over slightly ungrammatical ones. Moreover, the more errors in the sentence analysis, the less likely it becomes. Thus, instead of using the same penalty for the first or the tenth error in a sentence analysis, there should be, at least, a non linear progression of the penalty rate.

Taking the size of the sentence into account is perhaps the easiest aspect of the metric. The size of a sentence can be assessed by the number of words it contains. Thus, the number of errors detected can be normalized by the size of the sentence, by simply dividing it by the number of words. Therefore, a short sentence containing one error would have the same partial score as a sentence twice as long containing two errors.

Error categories have different frequencies. Thus, errors in the choice of the auxiliary are much less frequent than gender errors.[16] We use this knowledge to tailor our analysis selection metric based on errors by incorporating the relative frequency of errors categories within our formula. This implies the availability of an error tagged corpus to derive the correct frequencies, in order not to use handmade heuristics based on assumptions which could reveal faulty. Because more frequent errors should be less penalized, we need an inverted metric. One proposal would be to compute a kind of reverse

[16]This knowledge is derived from analyses of the FRIDA corpus (4.2.1).

percentage of the error. Typically, this would be $(100 - n)$, where n is the percentage of the particular error category. However, this would give scores for all errors between 90 and 100 as an analysis of the FRIDA corpus (4.2.1) gives all error category percentages between 0 and 10. Thus, the difference between two error categories would be relatively small. In order to circumvent this problem, we propose to use 10 instead of 100 in the formula, which would then become $(10 - n)$. In this way, the scores would vary between 0 and 10, thus greatly favoring frequent errors over rare ones.

In order to combine the three elements of the metric mentioned above, we propose the following formula where e corresponds to the errors in the sentence, $|e|$ being the number of errors. f is a function returning the frequency in percent of a particular error category. $|w|$ is the number of words in the sentence.

$$(1 + \frac{|e|^2}{10}) \cdot \frac{\sum_{i=1}^{|e|}(10 - f(e_i))}{|w|}$$

$(1 + \frac{|e|^2}{10})$ ensures that a great number of errors is less likely than a small number. Squaring the number of errors ensures a non linear progression. The square progression in tempered by the $\frac{1}{10}$ factor, and the addition of 1 prevents a null result. Division by $|w|$, the number of words, ensures normalization. Finally, $(10 - f(e_i))$ takes into account the frequency of error categories. In this optic, the higher the score, the less likely the structure.

3.4.3 Word frequencies

To evaluate the likelihood of a given analysis, one can also take word frequencies into account. When there are several possible analyses for a given input, it often happens that some ambiguous words are used with different meanings and/or lexical categories. Assessing the meaning of a polysemous word in context is far beyond the scope of this dissertation. Each particular analysis, however, must have selected only one lexical category per word. It is thus much easier to use lexical categories to separate homographs. If several lexical categories can fit within the syntactic structure, then different analyses are provided by non deterministic parsers.

We can try to evaluate the likelihood of a specific analysis against another one by looking at the relative frequency of the individual words used in the sentence, discriminating them by their lexical category. In its simplest version, the metric adds all the frequencies of the word-category pairs for each sentence structure version. The syntactic structure with the highest result is considered the most likely.

(43)a. *J'ai vu le problème.*

b. $[_{\texttt{TP}}\ [_{\texttt{DP}}$ j' $]\ [_{\overline{\texttt{T}}}$ ai $[_{\texttt{VP}}$ vu $[_{\texttt{DP}}$ le $[_{\texttt{NP}}$ problème $]\ ]\ ]\ ]\ ]$
I have seen the problem.

c. * [$_{\texttt{TP}}$ [$_{\texttt{DP}}$ j'] [$_{\overline{\texttt{T}}}$ ai [$_{\texttt{VP}}$ [$_{\texttt{AdvP}}$ [$_{\texttt{PP}}$ vu [$_{\texttt{DP}}$ le [$_{\texttt{NP}}$ problème]]]]]]]
I have in-view-of the problem.

In example (43), the parser provides two different analyses for the sentence. The first analysis, given in (43b), is much more likely than the second, presented in (43c). We can discriminate between the two and choose (43b) by adding word frequencies according to lexical category for each word of the sentence. As in this particular case only the word *vu* (see/in-view-of) is ambiguous for lexical category, it will be the only discriminating element. Corpora studies tell us, but it is also obvious for native speakers of French, that *vu* as a verb is much more frequent than as a preposition. Thus, (43b) receives a higher score than (43c) and is chosen as the best analysis.

Word frequencies depend on the corpus from which they were extracted. It is important to use, as much as possible, a corpus representative of the users of the system. Moreover, the larger the corpus, the better for the reliability of the frequencies. At first sight, one could propose to use a learner corpus for this task, as it would correspond to the users exactly. However, one does not want to use an error-full corpus, in order to avoid introducing non-word frequencies into the database. Moreover, the corpus needs to be tagged for parts of speech (or lexical categories) which is a very tedious work to do by hand. It can be done relatively reliably by a parser/tagger, but only if the input is grammatically correct. As this is not likely to be the case with learner corpora, it proves uninteresting to extract word frequencies from learner corpora without manually correcting these corpora. Thus, a corpus containing only grammatical sentences made of words from a fundamental lexicon seems more appropriate. A POS tagger will be able to establish the lexical categories of the words more easily. One could argue that some words will not appear in a fundamental lexicon. But, given the CALL context, this should not be too much of an issue, especially if one considers that a word not included in the fundamental lexicon is, by definition, rarer than any of the words included in it. Words not included in the corpus, and thus in the frequency lexicon, could simply receive a frequency value of 0, or even a negative number if one wanted to penalize them more heavily.

We have written so far about word frequencies without differentiating between inflected word forms and lexemes.[17] Frequencies can be calculated for both and thus both can be used in a metric for selecting the best analysis produced by a parser. However, results using inflected word frequencies or lexeme frequencies are not necessarily identical.

Let us note, first of all, that it is not always possible to discriminate between two analyses using only inflected word frequencies.

[17]A lexeme regroups all inflected word forms of a given word. A lexeme is often represented by the canonic form of the word, i.e. the infinitive for a verb.

(44)a. *Le fils de la tisserande.*

b. * [$_{\texttt{DP}}$ le [$_{\texttt{NP}}$ fils [$_{\texttt{PP}}$ de [$_{\texttt{DP}}$ la [$_{\texttt{NP}}$ tisserande]]]]]
the-singular threads of the weaver
Number error between *le* and *fils*.

c. [$_{\texttt{DP}}$ le [$_{\texttt{NP}}$ fils [$_{\texttt{PP}}$ de [$_{\texttt{DP}}$ la [$_{\texttt{NP}}$ tisserande]]]]]
the son of the weaver

In example (44), the sentence has two readings, an ungrammatical one (44b) and a grammatical one (44c). As in both analyses *fils* (threads/son) bears the lexical category 'noun', looking up the frequency of the inflected word form *fils* returns only one value and it is impossible to tell whether we are dealing with a 'thread' in the plural, or a 'son'. However, if we use lexeme frequencies, we must consider the lexemes *fil* (thread) and *fils* (son) which almost certainly do not have the same frequency. Thus, basing oneself on lexeme frequencies alone, and *fils* being more frequent, we select example (44c) as the best analysis. By chance, it corresponds, in this particular case, to the error-free reading.

Using lexeme frequencies instead of inflected word frequencies is also a manner to reduce difficulties linked to data not appearing in the corpus from which the frequencies were extracted. Most lexemes regroup several inflected words, so that inflected words which are not part of the initial corpus can find themselves covered by the lexemes from the same corpus. Thus, the problem of the unavoidable scarcity of the input data is somewhat reduced by the use of lexeme frequencies.

There might be cases, however, where using lexeme frequencies over inflected word frequencies is a disadvantage. Let us imagine the case of an inflected word which can belong to two (or more) lexical categories, say verb and noun. It is possible that the verb lexeme is much more frequent than the noun lexeme, in which case the verb reading would be selected over the noun reading when lexeme frequencies are used. It may happen, nevertheless, that the frequency of the specific form of the inflected verb is much less frequent than its lexeme. It might even be less frequent than the noun reading which is subdivided into only two inflected forms. *joues* (play-2nd-singular/cheeks) is a perfect example. Its verb lexeme, *jouer* (play), has a frequency of 505,[18] while the noun lexeme (cheek) has a frequency of 85. However, the second person singular form of the verb is much less used (frequency of 0) and certainly less frequently than the noun *joues* (cheeks) in its plural form (frequency of 26). We would thus favor the opposite reading by using either lexeme or inflected word frequencies.

[18]The numbers come from the frequencies assigned to each word form and lexeme in the Fips lexicon database.

Combining inflected forms and lexeme frequencies, by simply adding both, for example, might be an interesting solution, especially if both frequency tables are extracted from the same corpus.[19] Moreover, using lexeme frequencies can help selecting the appropriate lexeme when there are several lexemes of the same lexical category, such as for verbs with different subcategorization frames. In practice, using both types of frequencies, or only one, and which one, depends on the corpus available and on what can be extracted from it given the tools one disposes of, or on whatever frequency lexicons are available.

3.4.4 Combining the scores

To form a complete analysis selection metric, one must combine scores coming from at least three origins. One must normalize the scores in order for their values to be in the same range when they provide the same judgement. One could compute the average score of each component of the metric on the same set of sentences, and divide the individual results by this average. Combining the normalized scores by adding them would not work because, as we have described it above, low scores for error frequencies are good, while high scores are preferred for word frequencies. Syntactic likelihood scores depend on the parser used. We propose adding partial scores for which high values are preferred (word frequencies and syntactic likelihood) and subtracting those for which low values are needed (error frequencies). Only experiments could tell whether some weights on individual scores are needed, or if each component of the final metric has the same importance.

3.5 Combining techniques

Having several techniques for syntactic error diagnosis at disposal is an advantage only if the results from these different techniques can be employed in a coherent manner. Results provided by different techniques can converge or diverge on the diagnosis of particular errors. One must combine the results in order to reinforce converging diagnosis hypotheses and to be more careful with diverging results to minimize overflagging.

Several ways to combine diagnosis techniques and to manage their results are proposed below. We detail an approach in which each technique is independent from the others in section 3.5.1. We then discuss a cascade activation of the diagnosis techniques in section 3.5.2. We finally propose to choose the most relevant technique to be applied through heuristics in section 3.5.3. We conclude with some remarks (3.5.4).

[19] Otherwise, one must be careful to use only relative frequencies to account for a possible difference in corpus length.

3.5.1 Independent approach

The independent approach makes reference to all technique combinations in which diagnosis techniques do not depend on each other to be activated. Thus, in this approach, the techniques can be activated in any order. Moreover, they can be activated in a sequential or a parallel manner. This should have no influence on the final diagnosis result as one combines results from all the techniques together.

For this approach, one must manage results coming from different sources. These results can converge, that is, point all to the same error in the same location, or diverge on the presence of an error or on its category. Converging results on all techniques reinforce the diagnosis. Diverging results must be taken more carefully.

We use the location of an error as a basis to state the diverging possibilities. We consider that we have four groups of diagnosis techniques. Group 1 diagnosis techniques have detected an error of a specific category at a specific location. Group 2 techniques are concerned by the error category but have not found any error at that location. Group 3 techniques, also concerned by the error category, have found an error at the location, but it is of a different type. Group 4 techniques cannot treat errors of the specific category and are thus not taken into account for the particular error.

The possibilities, excluding group 4 techniques, are summarized in Table 3.1.

Table 3.1: Combining techniques

Situation	Group 1	Group 2	Group 3	Result
1	Error located	No error found	Other category	Warning
2	Error located	No error found	Group empty	Warning
3	Error located	Group empty	Other category	Warning
4	Error located	Group empty	Group empty	Error

Situation 1, where an error of a given type was detected only by some techniques, other techniques either not detecting any error or detecting errors of other categories at that location, produces a warning message indicating that there is a possible error at the location. The category of the error is unsure and the list of categories detected by techniques in groups 1 and 3 are listed.

Situation 2, where an error of a given type was detected only by some techniques, produces a warning as to whether there is indeed an error at the location. The error category is the one indicated by group 1 techniques.

Situation 3, where an error of a given type was detected only by some techniques, other techniques detecting errors of other categories at that location, produces a warning explaining that the error at the location is of different possible categories. The category list is provided by techniques from groups 1 and 3.

Situation 4 is the clearest. All indications are in favor of the same error category at the same location. The diagnosis is reinforced and an error message is output.

If warnings are not a possible option of the diagnosis system, instead of marking all possible errors which would risk increasing the rate of overflagging, one could indicate only errors for which group 1 represents the majority of the concerned techniques. That is, the number of techniques in group 1 must be at least as great as the number of techniques in groups 2 and 3. When dealing with only three techniques, as described in this dissertation, it comes to flagging errors in situation 2 if group 2 contains only one technique, in situation 3 if group 3 contains only one technique, in all the cases in situation 4, and never in situation 1.

Another possibility to combine results from different techniques is to decide that each technique is specialized for a specific set of error categories and that these sets are mutually exclusive. Thus, combining results from the different techniques simply means putting together all the diagnoses.

3.5.2 Cascade approach

The cascade approach is a sequential method in which techniques are activated one after the other depending on the results obtained by the previous technique. We must define the kind of results which activates the following technique and the order in which techniques must be activated.

A possible reason for a technique to be activated is if the preceding technique was not able to detect any error. It is the cue to continue looking for errors with other techniques, especially if one has the means to know in advance that the sentence is not grammatical, for example if it was not recognized by the parser in its strictest configuration.

One could also consider that the next technique should be activated if the previous one has not provided a full analysis of the sentence under treatment. This obviously presupposes that the previous technique can and does provide a full analysis if it has been able to detect all the errors present in a given sentence. If this option is selected, then combining different results is also an issue and can be resolved in the manner described for the independent approach in section 3.5.1.

Because it is possible that not all techniques are activated for each sentence, the techniques must be ordered in a specific manner. To do this, and because not all techniques detect all error categories, one should classify the error categories by the priority in which they should be detected. This priority may be derived from learner corpus analysis. Then, the techniques can be ordered according to the number of priority error categories they are built to detect. This would enable the full system to detect the most important errors with the first technique, thus perhaps achieving a full analysis by activating only one diagnosis technique and therefore minimizing processing time.

3.5.3 Heuristic approach

The heuristic approach is based on the fact that all three techniques described in this chapter can start by using a parser as a recognizer to filter out grammatical sentences which do not need to undergo error detection. Only if partial analyses are produced does one start the error detection process. Information can be gathered from the partial analyses to enable heuristics to point us to the most interesting diagnosis technique to be used. Only one technique would be selected by these heuristics, thus avoiding the difficulties of managing diverging results.

The advantage of the heuristic approach is that only one diagnosis technique is ever activated. This reduces diagnosis processing time. Moreover, the technique selected should, *a priori*, be the best suited to detect error categories present in the sentence, given the fact that the heuristics for technique selection take into account the information extracted from partial analyses of the sentence. Disadvantages of this approach include the complexity of deciding which particular technique to use in a given situation and the possibility that fewer errors will be detected because only one technique is used.

3.5.4 Remarks

We favor the cascade approach over the other two. The cascade approach seems much easier to implement than the heuristics approach for which it is unclear exactly what is the pertinent information which needs to be extracted from the partial analyses in order to select the relevant diagnosis technique, or how to extract this information. Compared to the independent approach, we hope for a decrease in diagnosis processing time with the cascade approach, as not all techniques will be activated on all sentences. Moreover, if one cascades to the next technique only when a full analysis has not been reached, new diagnoses should reflect only errors undetected so far (and located at the boundaries of partial analyses). Therefore, combining results from different techniques should be limited to putting together all the separate diagnoses obtained from the cascaded diagnosis techniques.

3.6 Conclusion

We have described three error diagnosis techniques in this chapter. They have varying advantages and disadvantages, depending partly on the parser they are implemented with. They do not cover the same ranges of error types. It seems important to compare and contrast them to evaluate which is the most adequate technique given a specific context and how to use them in general configurations. Table 3.2 gives a general overview of the different error types which can be detected by the three error diagnosis techniques. A '+' indicates that the technique is appropriate, a '0' shows a neutral position, and a '-' states that the techniques is not adequate for the particular error type. The error types are essentially drawn from the typologies established on the FRIDA corpus (4.2.1).

Table 3.2: Comparison of the error diagnosis techniques

	Constraint relaxation	Phonological reinterpretation	Chunk reinterpretation
Gender	+	0	+
Number	+	0	+
Person	+	0	+
Punctuation	-	0	0
Word order	0	-	0
Choice of auxiliary	0	-	+
Homophones	-	+	-
Class	-	-	0
Euphony	+	-	0
Verb complementation	+	-	+
Voice	0	-	-
Negation	+	-	-
Redundancy	-	-	0
Agglutination	-	+	-
Auxiliary and past participle	0	0	+

From Table 3.2, we can see at a glance that not all techniques are capable of diagnosing the same error types. All of them can diagnose at least some errors of agreement in gender, number or person, but only constraint relaxation treats negation errors. Homophones are only treated by phonological reinterpretation. Wrong forms for the structure 'auxiliary + past participle' are best detected by chunk reinterpretation. Although there is some overlap between the diagnosis techniques, we can see a certain complementarity between them. Thus, if one needs to detect only euphony errors for a particular application, the choice of the technique will rest on constraint relaxation.

For a general error diagnosis system, however, no technique is able to treat all error types. It seems therefore necessary to combine the techniques to propose the best overall diagnosis. Technique combination has been suggested and employed before (see Cornu (1997), Sanders and Sanders (1989), and Felshin (1995), among others), although most often the combination was between error grammars and constraint relaxation and each technique treated a different set of error types. Moreover, by combining several techniques, we can decide in advance which technique will be used for a particular error type and, in doing so, select error types with regards to their facility of treatment within a particular technique. This would help us to avoid many implementation difficulties which are encountered only when one starts to treat specific error types. These specific error types would then be taken care of by a more adequate technique.

In view of combining techniques together, we have proposed three approaches. As stated in section 3.5.4, we favor the cascade approach to diagnosis technique combination. We propose to cascade from constraint relaxation to phonological reinterpretation, and from then to chunk reinterpretation. This order is derived from the fact that constraint relaxation does not require chunks from a prior analysis as input. This technique can therefore be used first. Chunk reinterpretation does not necessarily attempt to recombine chunks to reach a full analysis. Thus, we want to use this technique only if the other ones were not able to provide a full analysis. It is therefore placed in last position. Phonological reinterpretation strands somewhere in between. It requires chunks as input and it provides, whenever possible, a full analysis. It is therefore sandwiched between constraint relaxation and chunk reinterpretation. Because constraint relaxation and phonological reinterpretation propose chunks only for errors they have not been able to diagnose, and because phonological reinterpretation and chunk reinterpretation can diagnose errors only around chunks, only new errors will be diagnosed by each successive technique. There should therefore be no competition between alternative diagnoses, even if several techniques are tailored to detect the same error types.

Two of the techniques discussed above, constraint relaxation (3.1) and phonological reinterpretation (3.2), strongly play with the non deterministic nature of the parser to reach their diagnosis and provide full analyses of their input. We have thus devised alternative ranking methods to supplement those normally included within a non deterministic parser. Ranking alternatives depending on the detected errors and of the word frequencies taken from a fundamental dictionary are specifically designed for error diagnosis within a CALL context. They would not be applicable as such with a more generic parser. Their effectiveness, however, still needs to be assessed.

This chapter exposed three error diagnosis techniques from a theoretical standpoint. Because many more full analyses can be provided when parsing with error diagnosis techniques, new methods for selecting the most likely analysis were designed. Several approaches for combining the three techniques were proposed and a preference was clearly indicated. Discussion remained mostly at the theoretical level. The error diagnosis techniques, scoring methods, and combination approaches must now be implemented and evaluated. These two points are the topic of the next two chapters. Chapter 4 proposes and discusses a specific implementation of the theoretical proposals of the present chapter, while chapter 5 evaluates the results of the implementation.

Chapter 4

Implementation phase

This chapter describes an implementation of the diagnosis techniques exposed in chapter 3. This implementation was realized by a team of researchers in the Department of Linguistics of the *Université de Genève*. The available resources did not allow us to create our own parser, thus it was clear from the beginning that we would have to reuse and transform an existing parser into a grammar error diagnosis system. The choice naturally rested on Fips, a large coverage parser for French, to which we had full access. Implementation of the diagnosis techniques had to take into account the specifics of the Fips parser.

This chapter is organized as follows. Section 4.1 describes Fips, the parser which is being transformed into an error diagnosis system. Section 4.2 discusses the empirical data which were used to design specifications for the system and to test it. Section 4.3 relates how each of the diagnosis techniques has been actually implemented and how the techniques' results are combined. Section 4.4 concludes this chapter.

4.1 Description of the Fips parser

The Fips parser has been described in many other places, including (Laenzlinger and Wehrli 1991), (Wehrli 1997), and (Laenzlinger 1998b). The goal of this section is not therefore to give a complete description of Fips, but rather to underline aspects of it which are important for syntactic error diagnosis. In this section, we discuss the main features of the linguistic theory Fips is based upon (4.1.1). We then provide information on the parsing algorithm which is used (4.1.2). The way Fips takes advantage of the implementation language is reviewed (4.1.3). Advantages and disadvantages of using the Fips parser in particular to transform it into a diagnosis system are exposed (4.1.4).

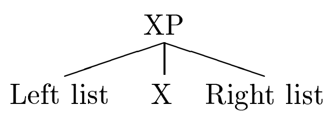
4.1.1 Linguistic theory

The government and binding (GB) theory has been first developed by Chomsky (1981) and subsequently elaborated and described by many, including Haegeman (1991). It is modular in that it posits the existence of a universal grammar, common to all languages, and language specific parameters. "For a variety of reasons, GB theory is not at this time the basis for many parsers, either within the domain of CALI or outside. Most parsers which use a linguistic theory are based on LFG or GPSG" (Bailin 1988, p. 30). Arguments in favor of using a GB family framework for parsing, in particular in the case of CALL, have nevertheless been put forth by Matthews (1993) and Weinberg et al. (1995). GB has been chosen as the underlying linguistic theory for the Fips parser which uses an adapted and simplified version of it.

In the implementation of the GB theory in the Fips parser, components of the grammar are translated as processes. There are structure generating processes, as well as filters which prevent the appearance of ungrammatical structures. "Certains [processus] sont générateurs de structures, comme le processus $\overline{\mathrm{X}}$ qui traite de la construction des chaînes $\overline{\mathrm{A}}$ et des chaînes clitiques, ou encore le processus de traitement de la coordination ; d'autres exercent une fonction de filtre, comme le filtrage des cas ou le module chargé de l'assignation et de la validation des fonctions thématiques" (Wehrli 1997, p. 214).

The main component of the system is the $\overline{\mathrm{X}}$ module which generates structures on the basis of the lexical elements. For each lexical element, a projection of the same lexical category is created with the lexical element as its head and two lists of projections, one for specifiers (Left list) and one for complements and adjuncts (Right list), both empty at first, attached to it. (45) gives a representation of a projection created on the basis of a lexical element of type X.

(45)



A maximal projection (level XP) can combine to another projection as its specifier (left list in French and English) or its complement (right list in French and English) if it fulfills the generic principles and the language specific parameters associated to the particular attachment. These are translated as constraints in the Fips parser.

4.1.2 Parsing algorithm

The syntactic parsing strategy chosen for Fips is that of the attachment to the right corner. Here are the four main principles of this strategy, taken from (Wehrli 1997, p. 220):

- Ascending strategy, that is data driven.
- Iteratively, from left to right, one reads a new element and one tries to combine it to a maximally developed structure from the left context.
- The left structure specifies a list of active nodes on which new elements can attach.
- All possible attachments are considered in parallel.

A lexical analysis is first performed to segment the input. For each lexical element, a projection of the same category is created. They are stored in a graph of well-formed (although possibly partial) constituents. The parser tries to combine two adjacent projections, a and b. a can attach as the specifier of b (left attachment), or b as a complement of an active node of a (right attachment). An active node is a position on the right edge of a tree structure which is available to receive new complements. The list of active nodes is kept and updated at each new attachment in order to facilitate and speed up the process of right attachment. Each new combination is stored in the graph. Possible attachments are treated in parallel, with the help of heuristics to restrain the size of the set of constituents to be considered for attachment. The output of the parser is a tree-like syntactic structure represented linearly by labeled brackets. In case no complete analysis is found by the parser, it returns partial analyses which, put side by side, cover the whole sentence.

4.1.3 Implementation language

Fips is currently implemented in Component Pascal, a dialect of Oberon-2 (Reiser and Wirth 1992, Mössenböck 1993). It is a mixed language, both procedural and object-oriented, and a descendant of the Pascal family. The implementation makes use of the object-oriented features of the language. Fips is in fact the French component of a family of parsers. The core parser contains the generic methods for all languages. It represents the universal grammar of the GB theory. Language specific modules implement methods which are related to language specific parameters. Types are extended from the language generic base types to take into account specific language features, such as clitic pronouns for French. Likewise, methods encoding constraints designed at the generic level can be redefined for every language when language differences must be taken into account. New methods are created for language specific constraints. A further step is taken in the direction of object oriented design by considering that error diagnosis is an extension of the types and methods defined for French.

4.1.4 Advantages and disadvantages

"Most parsers and other NLP programs are not typically designed to identify and diagnose errors; they simply fail when they encounter ill-formed input and do not identify the cause of the failure. For ICALI applications which require exact diagnosis of errors, parsing programs must be augmented with error detection procedures" (Bailin and Levin 1989, p. 6). This is naturally the case for Fips and some parsers might lend themselves more easily than others to the transformation into an error diagnosis system. We now describe some of the advantages and disadvantages of the Fips parser for this task.

Fips uses a chart, or graph, to store well formed constituents in order for them to be computed only once during analysis. This chart is also very useful when no full analysis is reached for a given sentence. All the constituents which were computed can be found in the chart. Therefore, the chart already contains the partial analyses which are to be displayed to the users. One must only retrieve from the chart what one considers the best series of chunks covering the whole sentence. No other computation is needed. As chunks are essential for both phonological reinterpretation and chunk reinterpretation, it is necessary to be able to retrieve them easily.

Non determinism is also a distinct advantage. It greatly facilitates the process of phonological reinterpretation. Reinterpreted words can simply be added as new alternatives and the parser tries to analyze the sentence with the reinterpreted words as well as with the original ones. Sentences which receive a full analysis with a reinterpreted word provide both the diagnosis and a correction proposal.

Constraints are an integral part of Fips and of the linguistic theory it is based on. This facilitates the task for constraint relaxation as constraints already exist. Nevertheless, the grammar is completely imbedded within Fips, which makes it difficult to pinpoint the precise location of some of the constraints, or of some of the conditions. Moreover, as Fips was not at first intended for error diagnosis, it is underconstrained in many ways and part of the transformation work consists in adding constraints appropriately, so that ungrammatical sentences do not pass unnoticed anymore. This is somewhat more of a challenge with a grammar which is not clearly separated from the parsing algorithm.

Another disadvantage or difficulty linked with Fips is that the parser is currently used for other applications. Thus, any modification which is made to the parser for error diagnosis must be transparent to the other applications. This implies that even restricting the parser for it to recognize only grammatical sentences cannot be done in a straightforward manner.

Modifications of the code must have an effect only when in the specific diagnosis mode; this complexifies the implementation of the modifications but some of the burden is fortunately taken care of through object-oriented programming.

Disadvantages of Fips thankfully do not preclude its great advantages: its robustness, its wide coverage, and its rich lexicon. They are crucial in building a real-life diagnosis system.

4.2 Errors to be diagnosed

An important step in the implementation of an error diagnosis system is to decide which types of errors are to be diagnosed. Realistically, not every imaginable error type can be diagnosed within a single system. First of all, we must remember that the diagnosis system presented in this dissertation concentrates on grammar errors, to the exclusion of spelling errors and errors linked to the semantics, pragmatic, or discourse levels.

Two main criteria impose themselves on the selection of error types for diagnosis. On the one hand, there is what is possible, and more or less easy, to implement given the parser at our disposal and the diagnosis techniques available. On the other hand, there are the needs of the end user population which makes specific kinds of errors. "Thus, 'easy to parse and still unstable' became a negotiated criterion for error inclusion, as did 'parser-amenable and consistently difficult.' These criteria helped to define a set of target errors that proved relevant to observed errors during user testing" (Holland 1994, p. 235).

In the remainder of this section, we describe a learner corpus, FRIDA, which is used to ascertain the needs of the target user population, and we make some observations on it (4.2.1). We then explain how error types were selected, based on several factors including knowledge of the parser and of the diagnosis techniques, the corpus, and pedagogical considerations (4.2.2). Next, we display an error example for each of the selected categories (4.2.3). Finally, we propose a data structure to encode the error diagnoses (4.2.4).

4.2.1 The FRIDA corpus

The French Interlanguage Database (FRIDA)[1] is a French learner corpus which was collected by the CECL.[2] The texts come from essays of intermediate learners of French from around the world, made available by corresponding teachers. These texts are often hand written and the first task

[1]For more information on the FRIDA corpus, please consult its web site (last accessed on April 8, 2003):
http://www.fltr.ucl.ac.be/fltr/germ/etan/cecl/Cecl-Projects/Frida/gateway.htm

[2]Centre for English Corpus Linguistics, *Université Catholique de Louvain.*

of the CECL has been to key them in, a lengthy and tedious process. Once in electronic format, the corpus has been manually tagged for errors. Error tagging is a necessary step before reliable error analyses can be performed on the corpus. The total size of the corpus approximates 450'000 words, of which a little over 300'000 are error tagged.

Errors are tagged using a triplet of XML tags opening on the left and closing on the right of an error. A correction proposal is also present. A triplet of tags is used in order to indicate the domain of the error (such as grammar), its more precise error category (number, for example), and the grammatical category of the erroneous word (e.g. tensed verb), as in example (46) taken from (Granger et al. 2001, p. 612).

(46)a. **Ils mange des pommes.*
They eats indefinite-article apples.

b. *Ils* <G><NBR><VSC> #*mangent*$ *mange* </VSC></NBR></G> *des pommes.*

A concordance tool is used to search the corpus along these tags. "On peut chercher les erreurs par domaine d'erreurs, par catégorie d'erreurs ou par catégorie grammaticale. Il est en outre possible de combiner ces différents types de recherche" (Granger et al. 2001, p. 612). Analyses performed by the CECL on FRIDA focus either on a particular grammatical construction or on a specific error category. Analyses per error category prove more interesting to error diagnosis because the correlation of the inner workings of the parser is greater with error categories than with grammatical construction. A first type of information which is provided on the FRIDA corpus, thanks to its tags, is the repartition of the errors by error domain, error category, grammatical category, as well as by triplet of XML tags. This repartition is given in percentages and it is thus easy to classify errors according to their relative frequency. These statistics provide the first information about where learners make more mistakes and thus about what the diagnosis system should focus on.

The error typology established by the CECL on the learner corpus distinguishes no less than thirty-six error categories. They range from errors of graphical form (e.g. missing accents), to errors of style (e.g. heaviness), via grammar errors (e.g. gender) and lexical errors (e.g. meaning). Obviously, not all of these errors should be treated by a grammar checker. Some are in the realm of a spell checker, others should be detected by tools such as semantic or style checkers. The limit between what belongs to grammar and what belongs to other levels is not always as clear cut as it may seem at first glance. For example, tense errors can be considered to be part of what a grammar checker should diagnose when a given tense or mode is required by the sentence structure, such as in example (47). It can also be a purely stylistic decision as in (48). It is considered that "l'incorporation d'un analyseur syntaxique [n'est] pas suffisante pour un phénomène aussi précis que la concordance des temps" (Selva and Chanier 2000, p. 411). We thus chose not to treat this phenomena within our grammar error checker.

(47)a. **Si j'**aurais** su, je serais resté.*
If I had-conditional-present known, I would-be stayed.

b. *Si j'avais su, je serais resté.*
If I had-indicative-imperfect known, I would-be stayed.

(48)a. *Soudain, l'orage a éclaté.*
Suddenly, the storm has broken.

b. *Soudain, l'orage éclata.*
Suddenly, the storm broke-preterite.

It is possible that only some errors within a given error category might be treated by the grammar checker component of a diagnosis tool. Moreover, error categories used in FRIDA are not necessarily homogeneous from the point of view of a parser-based error diagnosis system. A single error category may cover several phenomena that are treated in a distinct manner by a grammar error diagnosis system. Obviously, there is no one-to-one correspondence between error categories and constraints or the way Fips treats these phenomena. Thus, some categories can be partially treated only and detailed analyses of the error categories take all their importance as they highlight possible subtypes of error categories.

4.2.2 Selecting error categories

There are three main points which need to be taken into account when one makes a selection of error categories for syntactic diagnosis. They are the following:

1. availability of diagnosis techniques;
2. knowledge of target users' needs;
3. pedagogical considerations and context of use.

Diagnosis techniques

Each diagnosis technique has its own peculiarities which make it more adequate to treat some error categories over others. Moreover, some error categories cannot be diagnosed by any of the three techniques described in chapter 3. Thus the techniques which are employed in a diagnosis system should be considered at the time a decision is taken as to which error categories should be focused on.

We have so far settled on three different diagnosis techniques, namely constraint relaxation, phonological reinterpretation, and chunk reinterpretation. Constraint relaxation is often cited as the champion of agreement errors (Matthews 1993, Tschichold 1999). Other error categories which can be diagnosed by constraint relaxation are the choice of the proper auxiliary, some word order errors (when they are linked to lexical features), complementation errors, euphony errors, errors in the subtype of some lexical elements, and some negation errors.

Phonological reinterpretation is concerned with erroneous segments which sound like correct ones. Its field of action is thus much reduced compared to constraint relaxation. It is designed to diagnose the incorrect use of homophones, some agreement errors, and some punctuation difficulties. Used at the lexical level, phonological reinterpretation can help in proposing alternative correction proposals and in correctly splitting run-on words.

Chunk reinterpretation depends much on the set of rules associated with the implementation of the technique. It should be able to diagnose errors of agreement, euphony, and punctuation, as well as some errors of redundancy, complementation, and word order.

The set of errors mentioned for each category is probably only a sample of what can be diagnosed for each of them. It is thus not to be taken for an exhaustive list. It is rather the direction in which we have been working with these techniques.

Knowledge of the target users' needs

The FRIDA corpus contains texts written by members of our target population. Studying FRIDA thus gives us insight as to the users' needs. It informs us on the errors which are actually made by language learners. Only error categories actually found in the corpus need to be diagnosed as supposedly only those will be found in the learners' input. Moreover, some error categories are more frequent than others, giving us an idea as to the relative importance of some errors. We can logically assume that it is more important to diagnose frequent error categories over rare ones which might appear only once or twice in the whole corpus.

Pedagogical considerations and context of use

On pedagogical grounds, all errors should probably be detected, even if they are not all reported to the learners. This would at least ensure a proper knowledge of a learner's level if it were to be included in a student model. This is, however, not very realistic in an automatic diagnosis system and thus some priorities must be set.

Experts in didactics distinguish between error categories which must be treated absolutely, given the level of the target users, and other categories which are deemed less important. They might be less important because they are already well mastered by the users or because they are way beyond their current possibilities.

The context of employment of the the diagnosis system must also be taken into account. If it is designed for use with a general text processing system, nothing specific can be said with regards to difficulties that particular users would encounter. On the other hand, if it is part of a CALL software and used for the correction of exercises, it is important that the difficulties of the specific exercises be taken into account and that errors on these difficult areas be properly diagnosed. It would otherwise defeat the purpose of an error diagnosis system integrated into a CALL software.

4.2.3 Error categories and examples

A number of error categories have been selected for grammatical error diagnosis. They take into consideration the three points mentioned in section 4.2.2. Besides, we also recognize that errors are not necessarily independent from one another. "Because syntax is hierarchical, dependencies may arise among errors that are flagged by the parser and reported to the students" (Holland 1994, p. 246). This implies that related errors must all be detected together.

(49) **Ils avantage**s** leurs amis.*
They advantage-2nd-singular their friends.
Number and person errors between the subject and the verb.

For example, when a person error occurs on a verb, it is sometimes linked to a number error as well. In (49), the incorrect plural marker makes the parser recognize the verb as a second person singular, thus both a person error and a number error are diagnosed. If the system was set up to diagnose only one of the two categories (number, for example), the error of the other category (person) could prevent the diagnosis of the number error because the subject could still not attach to the tense phrase.

Table 4.1, on page 122, gives the list of selected error categories together with their abbreviation, a short description of the error category, the importance of the error category within the corpus and from a didactic viewpoint, and an indication as to which diagnosis technique is used to detect the error category. Examples for those error categories are given in Table 4.2, page 125. More explanations are given below for each category in a less synthetic form.

Table 4.1: Selected error categories

Error category	Description	Corpus	Didactics	Techniques		
AUX	Wrong choice of auxiliary	1	2	R		
CLA	Use of the wrong class of word (lexical category or subtype)	3	3	R		
CPA	Incorrect adjective complementation	1	3	R		C
CPV	Incorrect verbal complementation	2	3	R		C
EUF	Euphony error	2	1	R		C
GEN	Gender agreement error	3	3	R	P	C
HOM	Use of an incorrect homophone	2	1		P	
MAN	Missing element in the sentence	3	2	R		C
NBR	Number agreement error	3	3	R	P	C
NEG	Missing or superfluous negation element	?	2	R		
ORD	Wrong order of words	2	2	R		C
OUB	Missing punctuation	3	1		P	C
PER	Person agreement error	1	3	R	P	C
VOI	Incorrect use of a pronominal reading of a verb	1	1	R		

Legend:
Corpus and didactics interests in the error category range from 1 (lowest) to 3 (highest in importance).
R = Constraint relaxation
P = Phonological reinterpretation
C = Chunk reinterpretation

Auxiliary errors (AUX) concern the use of the wrong auxiliary with a given verb. Use of *avoir* (have) instead of *être* (be) is easily detected. The reverse is harder because *être* (be) is also used in passive constructions which are perfectly grammatical. One should thus flag an erroneous use of the *être* auxiliary only if one is not dealing with a passive structure.

Errors of class (CLA) indicate that a word of a wrong lexical category has been used (adjective instead of adverb, for example), or that the word does not belong to the expected subtype (e.g. different types of pronouns). A great number of the errors of the class category concerns the use of definite vs. indefinite determiners. This relates more to semantics and is not treated here. Although this error category is important for both corpus and didactics, few errors of this category can be detected by a purely syntactic diagnosis system.

Adjectival complementation errors (CPA) are supposedly errors in the preposition introducing the complement of an adjective. In the FRIDA corpus, though, many errors relating to the structure of adjectival predicates are tagged as such.

Verb complementation errors (CPV) relate to errors in the complements of verbs, their types, and the prepositions used. This is a difficult area for language learners as attested by both the corpus studies and the importance to this error category given by the didactic team.

Euphony errors (EUF) occur quite often but are not interesting from a didactic point of view. They are errors of elision and contraction. These phenomena are almost purely mechanical and do not hinder communication.

Gender errors (GEN) are frequent and important. They are sometimes linked to a poor knowledge of the gender of nouns, always a difficult topic in foreign languages. The error category is easily diagnosed by constraint relaxation. Phonological reinterpretation can also detect it if there is no sound change, and chunk reinterpretation might spot it if the two unmatched elements are in a good configuration.

Homonymy errors (HOM) concerns the use of words which sound alike and are wrongly interchanged. A diagnosis system based on a syntactic parser without semantics can only hope to detect homophones of different lexical categories. Our diagnosis is based on phonological reinterpretation.

Missing elements (MAN) are particularly hard to diagnose automatically as the error is in fact not there. It is however a frequent error category in the corpus and quite important didactically speaking.

Number errors (NBR) occur often and are important. Like gender errors, they can easily be diagnosed by constraint relaxation and possibly by the other two techniques.

Negation errors (NEG) are not coded as such in the corpus, which explains why we do not know its relative frequency. It concerns essentially the omission of the negative particle *ne* or of *pas* (not). While missing elements are involved, errors of negation are relatively easy to diagnose by constraint relaxation because two negative elements must be used as a pair.

Word order errors (ORD) concern adverbs (ORDAV) and adjectives (ORDAJ) as a priority, but also, to a lesser extent, verbs and pronouns. While it is possible to detect these errors with constraint relaxation, it is a known cause of overgeneration of structures. It is thus dangerous to relax word order too freely.

Missing punctuation errors (OUB) and in particular commas, are among the most frequent errors. However, punctuation is arguably a stylistic matter and rules governing it are not always very precise. Thus, this is a difficult category to diagnose. Full sentence reinterpretation is a theoretical option and it is also possible that chunk reinterpretation will detect some errors.

Person agreement errors (PER) are few in number, but they are linked to gender and number errors and should be detected at the same time.

Finally, voice errors (VOI) concern mainly the overuse, or lack of use, of the pronominal variant of a verb. It is infrequent and difficult to diagnose, apart from the flagrant cases in which an essentially pronominal verb is used without its reflexive.

Table 4.2 (page 125) provides an example for each of the selected categories in order to have a clearer view of the error categories we are talking about. Boldface type is used to highlight the erroneous parts for easy identification. A • indicates a missing element.

4.2.4 Error coding data structure

Once an error is detected in an input sentence, it must be recorded. To record an error, one must keep track of its type and its location. Difficulties start with the location of errors. Errors do not exist in and out themselves. They are linked to a context, a sentence. Moreover, errors often occur between two mismatched elements. This is particularly true with errors linked to constraints because, as we have seen it in section 3.1, constraints take two elements into account. If two elements do not match, it is impossible to decide automatically which of the two is incorrect. Probabilities and heuristics might help, but they are certainly not foolproof. Thus, we have decided to mark the errors on both mismatched elements whenever possible. An error can therefore have two parts, two components.

Errors are encoded at two levels. There is an internal representation of an error, as well as a representation for the purpose of display to the users. The internal representation uses a specific, extensible, type. In the current implementation, which is slightly simplified in example (50), it contains the error categories found at that location, divided into two fields. The first one is called 'diagnosis' and represents true errors of the syntactic level. The second, named 'warning', is similar to the first, except that it is used used for potential errors recognized by the diagnosis system with a lesser degree of reliability, or for structures which can be grammatical in specific

Table 4.2: Error categories and examples

Error category	Example
AUX	**Il* ***a*** *venu hier.*
	He has come yesterday.
CLA	****Ce*** *me plaît.*
	It me pleases.
CPA	**Il est* ***fier à*** *Marie.*
	He is proud to Marie.
CPV	**Il* ***la*** *parle.*
	It her-accusative talks.
EUF	**J'expose* ***le*** *argument principal.*
	I expose the argument main.
GEN	****Elle*** *est* ***venu*** *hier.*
	She has come-masculine yesterday.
HOM	**Il faut* ***maître*** *au monde des enfants.*
	One must master to-the world the children.
MAN	**Ils ont de • force.*
	They have some strength.
NBR	****Ils pose*** *les questions.*
	They asks the questions.
NEG	**C'• est pas un problème.*
	It is not a problem.
ORD	**Une* ***intelligente*** *femme.*
	An intelligent woman.
OUB	**Les pommes • les poires et les raisins.*
	The apples the pears and the grapes.
PER	****Je dort.***
	I sleeps.
VOI	**J'• évanouis.*
	I faint.

contexts but which are not very likely to be used by language learners. A set variable is used for both, as there can be several errors at the same location. Moreover, a boolean variable indicates whether the error is composed of one or several parts. In an array, an error index is given for each detected error category; this permits the system to link two parts of a single error together, if need be. This data structure is linked to a projection, which is taken as the location of (one part of) the error.

(50)

```
GrammarErrorPtr* = POINTER TO EXTENSIBLE RECORD
                          (FipsTools.GrammarErrorPtr)
    diagnosis* : SET;
    warning* : SET;
    errorIndex* : ARRAY ErrorTypes OF ARRAY 3 OF CHAR;
    manyPartError* : BOOLEAN;
END;
```

On the output end, the whole sentence structure is provided in an XML format, and thus the error marks are also included in this format. This is obviously an intermediary representation which must be translated into a user-friendly format before display to the users. This user-friendly display is not, however, part of our mandate and we limit ourselves to the XML output. We use two different XML tags. The first one, named `ERROR`, has four attributes: the error category, a single vs. many part indication, a warning vs. true error indication, and an error index. The second tag is entitled `PARTERROR` and corresponds to the second part of a multi-part error. Its only attribute is the error index linking it to the tag of the first part of the same error. The first tag contains the error category so there is no need to repeat this information as an attribute of a `PARTERROR` tag. Each error tag opens at the beginning of the error location (projection or word) and closes after it.

(51) *Il* <ERROR category="NBR" manypart="yes" warning="no" index="a0">*les* </ERROR> *a* <PARTERROR index="a0">*regardé*</PARTERROR> *attentivement.*
Il les a regardé attentivement.
He them has looked-at-singular attentively.

Example (51) presents XML tags indicating a number error between the preverbal clitic pronoun *les* (them), representing the direct object of the verb, and the past participle *regardé* (looked-at).[3] Only the error tags are shown, excluding structural tags, so as to simplify the representation.

[3]In French, a past participle conjugated with auxiliary *avoir* (have) must agree in

4.3 Implementation

An important consideration in the implementation of the error diagnosis system is that the Fips parser, which serves as a basis for the system, is concurrently used for other applications, such as speech synthesizing and machine translation. Thus, while transforming Fips to enable it to detect grammar errors, we have to maintain its previous functionalities and ensure that its output, when error diagnosis is not activated, remains the same as before the implementation of the diagnosis techniques.

The current section describes how the grammar error diagnosis techniques described in chapter 3, that is constraint relaxation, phonological reinterpretation, and chunk reinterpretation, were implemented (or not), respectively in sections 4.3.1, 4.3.2, and 4.3.3, taking into account the parser itself, the errors which have been selected for diagnosis, and the possibilities of each technique.[4]

4.3.1 Constraint relaxation

"GB's rules (principles) are expressed for the most part as well-formedness conditions (constraints) on linguistic forms" (Catt 1988, p. 19). As such, GB parsers are particularly well suited for modifications involving constraint relaxation. In Fips, the grammar is embedded within the code itself and there are no explicit rewrite rules. In place of these rewrite rules, possible attachment points are specified in terms of types of projection and types of attachment (specifier or complement). These correspond to the category constraints proposed in section 3.1.1. "Il va de soi que des contraintes additionnelles viennent limiter les possibilités d'attachement d'un constituant comme spécificateur d'une projection" (Wehrli 1997, p. 218). The same naturally applies to complements as well. The additional constraints are hard, soft and conditional constraints. They control constituent attachment to ideally allow only the construction of grammatical sentence structures. Each constraint violation is the mark of an ungrammaticality in the sentence. Thus, each time a constraint is relaxed, an error is detected.

We here describe the specific variant of the constraint relaxation technique which was implemented. We then discuss the problems which were encountered during the implementation phase. Finally, we expose the extent and limits of the current implementation and propose an informal assessment.

gender and number with its direct object when the latter is placed before the verb. This rule applies when a preverbal clitic pronoun stands for the direct object of the verb, as in example (51).

[4]It is to be noted that the author of this dissertation was not the main programmer in the team implementing the constraint relaxation technique in the error diagnosis system, although she wrote the specifications for it.

Variant employed

As we have explained above (4.2.3), not all errors are to be treated by constraint relaxation and not all constraints must be relaxed. There is a correlation between detectable error categories and constraint relaxation. The first step in the implementation of the constraint relaxation component of the diagnosis system is thus to decide which error categories are to be diagnosed with this technique. Once this is done, one must identify the constraints that correspond to the selected error categories. These two steps are in fact performed sequentially for each error category, rather than first selecting all the error categories and then finding all the corresponding constraints. One starts by selecting one error category which is to be diagnosed with constraint relaxation and then one identifies its corresponding constraints. More than one constraint often needs to be located in order to treat one error category completely. Once an error category is treated to the extent possible with constraint relaxation, a further error category is selected, and the process starts over again. As an example of an error category requiring the relaxation of several constraints, let us look at the gender error category. This category groups all errors of gender agreement, whether they occur in the noun phrase (52a), with a predicative adjective (52b), or with a past participle (52c). In Fips, this is represented by several constraints, one for each situation. Thus, there is (at least) one constraint for determiner-noun phrase agreement, one for subject-predicative adjective agreement, and one for past participle agreement.

(52)a. **Le maison.*
The-masculine house-feminine

b. **Elle est beau.*
She is beautiful-masculine

c. **La pomme que j'ai mangé.*
The apple-feminine which I have eaten-masculine

For each error category, one can either relax all the constraints linked to that error category, or relax only some of the constraints if they circumscribe an error subtype, if the other constraints are difficult to locate and/or to relax, or if relaxing the other constraints would be too disruptive to the inner workings of the parser (generating too many spurious analyses, for example).

We now turn to the manner in which constraints are relaxed in our diagnosis system. We have defined before that constraints are a set of conditions on which depend some actions of the parser (section 3.1.1) and that constraint relaxation changes the content of the actions linked to the constraints (section 3.1.5). Relaxing constraints in Fips implies first checking a number of conditions to circumscribe the precise situation in which a specific constraint may be relaxed. The constraint itself contains both a set of conditions as well as a set of actions depending on those conditions. The additional conditions and the actions are isolated into a procedure associated to a specific data type. There are in fact two such procedures for each constraint: one for diagnosis mode and one for non-diagnosis mode, as in the latter case Fips must keep its previous output. In non-diagnosis mode, the constraints are enforced and the actions performed correspond mostly to what Fips did before its transformation into a diagnosis system when constraints were unsatisfied: reject the attachment, which, in the long run, leads to outputting partial analyses.

The implementation of the constraint relaxation technique within the Fips parser makes full use of the fact that the parser is implemented in an object-oriented programming language.[5] Projections, which are the building blocks with which a sentence structure is constructed, are encoded as objects with associated procedures, called methods.[6] Depending on the dynamic type of the projections, different versions of the methods are called into play. The dynamic type of a projection is a function, at least in part, of whether syntactic error diagnosis is activated through an option of the parser. In case the error diagnosis option is selected, the type of the projection is extended with information about possible detected errors.

Constraints which can be relaxed are encapsulated into methods. For each constraint, there are (at least) two methods. One represents the conditions and actions for the constraint when it is enforced, the other when it is relaxed. Thus, different actions can be performed in each case. Modifying one of the methods does not influence its sister method in any way. The selection of which sister method to use in any particular case is automatically achieved through dynamic binding.[7] For example, on a gender agreement constraint when the two elements do not agree, the enforced constraint method returns an empty set and a Boolean negative value indicating an incorrect parsing path. In the same situation, the relaxed constraint returns a non-empty set resulting from the union of the gender sets of both elements and a positive Boolean value, indicating that parsing can continue.

The constraints in our system are not weighted. They are either enforced or relaxed. However, not all constraints need to have the same status. There is an internal device which allows the programmer to switch constraints on and off. This is implemented with a set for disabled relaxations of constraints which is initialized before parsing starts. Methods linked to disabled relax-

[5]"Object-Oriented programming means programming with abstract data types (classes) using inheritance and dynamic binding" (Mössenböck 1993, p. 6).

[6]For object-oriented terminology, see Table 12.1, page 238, in (Reiser and Wirth 1992).

[7]"This is what is meant by dynamic binding—an appropriate procedure is bound (called) at run-time by examining the dynamic type of the caller. That is, the specific action performed is determined at run-time" (Nikitin 1997, p. 175).

ations fail if their conditions are not met. Disabling specific relaxations of constraints is useful for two main reasons. (i) Implementation of relaxation by packets is facilitated by this device, which also allows for tests to be run in the way to allocate constraints to packets. We have, however, settled on a one pass system. Therefore, the set of disabled relaxations is empty during the first pass and all constraints are relaxed at once. (ii) In the testing phases also, it can be interesting to disable all but one of the constraints to see how it works in isolation. Thus, it has been possible to confirm that the adverb order constraint was a sure cause of much overgeneration of structures. To remedy this situation, we restricted the way in which the adverb order constraint was relaxed: the constraint relaxation was applied to fewer adverb types, and only to locations in the sentence where the corpora studies showed that learners made mistakes. For example, manner adverbs, such as *facilement* (easily), are allowed to attach with an error mark before the tensed verb of a sentence (52a), besides their legal positions (52b). However, attachment between the verb and a negation is never allowed, and (52c) does not receive a full analysis.

(53)a. **Je facilement ne vais pas au cinéma.*
I easily NEG go not to-the movie-theater.
Adverb order error on *facilement* (easily).

b. *Je ne vais pas facilement au cinéma.*
I NEG go not easily to-the movie-theater.

c. **Je ne vais facilement pas au cinéma.*
I NEG go easily not to-the movie-theater.
No full analysis of the sentence.

We have decided not to use graded constraints because they seemed difficult to implement within an already existing parser which possesses a scoring mechanism. The two systems might have interfered. Moreover, it is not clear how different diagnosis techniques can be used in conjunction with a graded constraint system. To take into consideration the relative importance of constraints, which seems an interesting point to us, we use instead a scoring mechanism (described in sections 3.4.2 and 4.3.4) in which frequencies of error categories are employed.

Problems encountered

Two main difficulties were encountered during the implementation of constraint relaxation on the Fips parser. The first one is linked to the fact that Fips was underconstrained in the first place. The second is inherent to constraint relaxation and concerns the overgeneration of structures.

In the description and definition of the constraint relaxation technique (section 3.1), we have always assumed that the parser, before it was transformed into a diagnosis system, acted as a recognizer and should still do so when all its constraints are enforced. This is unfortunately not the case with Fips. Fips was designed to treat grammatical sentences and no strict mechanism was implemented in it to reject all ungrammatical sentences. Fips is thus underconstrained in some areas and tries to find a syntactic structure for every sentence rather than being restrictive and prescriptive in what it accepts. At the same time, however, Fips requires some constraints to be enforced, such as euphony, agreement, case, and diacritics, in order to restrain the number of alternative paths to be investigated during parsing. The ungrammatical sentence in (54) with an incorrect pronoun case receives a full analysis from the untransformed Fips parser, while the one in (55), which requires only a simple elision to be correct, is rejected.

(54) **Je lui aide.*
I to-him/her help

(55) **Je aide.*
I-non-elided help

Thus, the first task we had to deal with was to reinforce Fips to bring it closer to a recognizer. For this, we added a number of new constraints in the parser. These new constraints, when they were violated, did not necessarily lead to the rejection of the structure, because the same structure was accepted by Fips in its standard version. Error marks were simply added to the structure, in order to be able to display the diagnosed errors to the learners.

Reinforcing a parser so that it acts more like a recognizer must by necessity be done either before or at the latest at the constraint relaxation stage in a diagnosis system combining constraint relaxation, phonological reinterpretation and chunk reinterpretation. Indeed, the latter two techniques use chunks as raw material while constraint relaxation can have an unparsed sentence as input. Thus, only the addition of relaxable constraints is able to diagnose errors which Fips used to let pass without noticing.

Inherent to constraint relaxation is the problem of the overgeneration of structures. By relaxing constraints, many more complete analyses can be found for a single sentence. As soon as there is more than one analysis, however, selecting the most appropriate analysis becomes an important problem. This difficulty naturally increases when the number of full analyses grows. "If the grammar becomes more complex, several competing parses for a given sentence might be found. Diagnosis then depends on what parse has been chosen" (Thurmair 1990, p. 365–366). Each analysis can bear different diagnoses because its structure is different or because several parsing paths were possible for the same sentence structure. As only one analysis is retrieved for display of the error diagnosis, one can easily see that it is crucial to choose this analysis appropriately in order to provide good diagnosis results.

With constraint relaxation as implemented in Fips, it often happens that the selected analysis does not bear the correct diagnosis for the sentence. However, the correct analysis with appropriate error diagnosis is part of the pool of complete analyses. For the sentence in (56a), the first analysis, given in (56b), does not provide an error diagnosis, while the second, in (56c), does.

(56)a. **Il y a pas longtemps.*
It there has not a-long-time

b. $[_{\text{TP}}\ [_{\text{DP}}\ \textit{il}\ ]\ [_{\overline{\text{T}}}\ \textit{y a}\ \ [_{\text{VP}}\ [_{\text{AdvP}}\ [_{\text{AdvP}}\ \textit{pas}\ ]\ [_{\text{Adv}}\ \textit{longtemps}\ ]\ ]\ ]\ ]\ ]$

c. $[_{\text{TP}}\ [_{\text{DP}}\ \textit{il}\ ]\ [_{\overline{\text{T}}}\ \textit{y a}\ \ [_{\text{VP}}\ [_{\text{AdvP}}\ \textit{pas}\ ]\ [_{\overline{\text{V}}}\ [_{\text{AdvP}}\ \textit{longtemps}\ ]\ ]\ ]\ ]\ ]$
Negation error on *a*.

Thus, the problem does not lie in the relaxation of constraints, but in the selection of the appropriate alternative. Although the system is working properly, it seems from the outside that its results are far from good. There are several partial solutions to this problem. The first one is to relax fewer constraints and/or to be stricter in the manner relaxation is allowed. This should decrease somewhat the number of superfluous analyses and therefore reduce the problem. This problem does not however disappear as long as there are more than one complete analysis. Another more general tentative solution is to formulate new ways to retrieve the most adequate analysis from the pool. This is attempted in the diagnosis system by taking into account both information about the errors detected (3.4.2) and word frequencies (3.4.3) as complementary information to retrieve the best analysis.

Extent and limitations of the implementation

Errors categories treated by constraint relaxation are quite numerous, but the extent to which each category is treated varies greatly from one category to the next. In order to discuss the extent and limitations of the implementation of constraint relaxation, we consider, in turn, each of the error categories of Table 4.1 (page 122) which were planned for treatment with constraint relaxation.

Employing the wrong auxiliary (category AUX) is well detected in several cases. Use of the *avoir* (have) auxiliary instead of *être* (be) is detected with predicative adjective structures, with pronominal verbs, and with other verbs which should be conjugated with the *être* (be) auxiliary. Verbs found with *être* (be) while they should conjugate with *avoir* (have) are harder to detect because of the possible confusion with the passive construction.[8] To prevent too much overflagging, we have limited ourselves to diagnose such errors for verbs which cannot take a passive form only; there is then no possibility to mistake the use of the *être* (be) auxiliary for a passive.

Diagnosis of class errors (CLA) is rather limited, compared to the definition of the FRIDA error tag. In the corpus, many CLA errors are semantic in nature, such as the choice between a definite or indefinite determiner, as in (58).

(58)a. *pour* ***une*** *continuation de l'espèce*
for a continuation of the species

b. *pour* ***la*** *continuation de l'espèce*
for the continuation of the species

This error subtype cannot be detected with a purely syntactic error diagnosis system. Therefore, we only detect two subtypes of CLA errors, namely the use of the pronoun *ce* (it) as subject of any verb apart from the copula, and some cases of the use of a strong pronoun (59a) when a weak

[8]Let us take an example to expose the risk of overflagging in the instance of the *être* (be) auxiliary being used erroneously.

(57)a. *Le chat a mangé la souris.*
The cat has eaten the mouse.

b. *Le chat a mangé.*
The cat has eaten.

c. *Le chat est mangé.*
The cat is eaten.

d. ?*Le chat est mangé par la souris.*
The cat is eaten by the mouse.

e. **Le chat est mangé la souris.*
The cat is eaten the mouse.

In example (57), sentences (57a) and (57b) are grammatical. They represent two perfectly normal ways to use the verb *manger* (eat) in the past tense. (57c) is either a legal passive or contains an auxiliary error. It is ambiguous and there is no way at this stage to determine which is the best interpretation and, therefore, one does not want to flag an auxiliary error. If (57c) was completed as in (57d), then the passive interpretation would have been correct. Note that (57d) is perfectly grammatical although semantically weird (hence the '?'). Finally, (57e) is without any doubt ungrammatical. It is however ambiguous as to the reason of the ungrammaticality. One can consider either that it is an auxiliary error (with (57a) as the correct form), or that there is a verb complementation error (with (57d) as the correct form). Because it is only when the parser reaches the word *la* (the-feminine) that an error can be detected, we have chosen the second option and indicated a verb complemetation error.

one alone is permitted,[9] as in example (59b).

(59)a. **Qui sont-eux?*
Who are-them?

b. *Qui sont-ils?*
Who are-they?

Although the corpora studies indicate both predicative structure errors as well as truly adjective complementation errors in the CPA category, the system only treats true errors of adjective complementation. Unfortunately, the well-known problem of the ambiguity of attachment of prepositional phrases often leads to overflagging occurrences in this error category. Thus, if the chosen (simplified) structure for (60a) is (60b) instead of the correct (60c), a CPA error will be marked on the structure.

(60)a. *Il est fier avec raison.*
He is proud with reason

b. $[_{\mathrm{TP}}$ $[_{\mathrm{DP}}$ *Il* $]$ $[_{\overline{\mathrm{T}}}$ *est* $[_{\mathrm{AP}}$ *fier* $[_{\mathrm{PP}}$ *avec raison* $]$ $]$ $]$ $]$
Adjective complementation error between *fier* (proud) and *avec* (with).

c. $[_{\mathrm{TP}}$ $[_{\mathrm{DP}}$ *Il* $]$ $[_{\overline{\mathrm{T}}}$ *est* $[_{\mathrm{AP}}$ *fier* $]$ $[_{\mathrm{PP}}$ *avec raison* $]$ $]$ $]$

Verb complementation errors (CPV) are hard to diagnose for two main reasons. (i) Verbs often have several lexemes with different subcategorization frames. It is difficult, not to say impossible, to decide which one to use when looking for verb complementation errors. We are reduced to checking that a particular argument is ungrammatical for all lexemes of the verb. (ii) The first difficulty is greatly compounded by the ambiguity of prepositional phrases which can often either be read as a complement or act as an adjunct. We currently diagnose errors on clausal complements, but only the omission of a preposition in nominal verb complements. Preposition confusion and addition in nominal complements are so far taken to be adjuncts and a complement to be lawfully missing. However, if we were to flag all adjuncts as potential complements with an incorrect preposition, we would drastically increase the number of overflagging occurrences.

Euphony errors (EUF) are very local errors, which make them stand apart from other error categories. The main difficulty for this error category was to derive an algorithm to treat linearly adjacent words within the hierarchical parsing strategy. Three subtypes of euphony errors are treated:

[9]No distinction is made here between weak and clitic pronouns. For a discussion of clitic, weak, and strong pronouns, see Laenzlinger (1998a, p. 125ff.).

preposition-determiner contractions, elision phenomena, and the euphonic *-t-*.[10]

Agreement errors (categories GEN, NBR and PER) are easily treated by constraint relaxation. They are most often checked by pairs with varying configurations depending on the lexical categories of the elements between which agreement is checked.

Missing elements in the sentence (MAN) are in fact not treated as such by constraint relaxation. Some missing elements are however diagnosed as part of another error category, such as verb complementation or negation.

Errors of negation (NEG) are diagnosed in relatively simple cases in which one part of the negation is present without its second element. Some cases in which *ne* (negative particle) can appear by itself are also taken into account. Distinctions between different negative adverbs (such as *pas* (not), *jamais* (never), and *plus* (more)) are not made in the grammar checker as they are related to semantics.

Three main subtypes of the order error category (ORD) are diagnosed: adverb order (ORDAV), adjective order (ORDAJ), and to a lesser degree verb order. The adverb order constraint is relaxed cautiously. Only some specific types of adverbs can attach to specific, ungrammatical positions, the types of adverbs and the positions being dictated by corpus studies. Blindly relaxing the adverb order constraint generated too many spurious analyses and occurrences of overflagging. Postnominal adjectives used prenominaly are marked with an error, while the reverse structure, prenominal adjectives found postnominaly, are marked with a warning only, as they are often, though not always, grammatical.[11]

The OUB error category, missing punctuation, was only partially implemented. The diagnosis systems detects only missing hyphens and has been trained even more specifically on missing hyphens between verbs and subjects in inverted interrogative sentences, such as in (63).

(63) **Sont ils partis?*
Are they gone?

[10] An example of the use of this euphonic *-t-* can be found in example (61).

(61) *A-**t**-il dormi?*
Has -t- he slept?

[11] (62a) and (62b) are both grammatical although they do not have the same meaning.

(62) a. *Un grand homme*
A great man

b. *Un homme grand*
A man tall

Error of the voice category (VOI) are treated only partially. Essentially pronominal verbs used without their reflexive are diagnosed, as well as non pronominal verbs which do not subcategorize for any complement but are used with a reflexive, as in (64).

(64) **La parité sociale commence à s'apparaître.*
The parity social starts to itself appear

This is far from treating all instances of this error category. However, as this type of error is not very frequent and of little didactic relevance (see Table 4.1, page 122), we decided not to spend any more time on it.

One of the limitations of the constraint relaxation technique concerns the lexicon used by the system. This technique requires one to conduct many tests on word features which are extracted from the lexicon. Incomplete or wrong information naturally leads to a decrease in performance of the diagnosis system. Although spot checking has been performed, we cannot be sure that all prenominal adjectives are listed as such in the lexicon, for example, nor that all possible subcategorization frames have been entered for a given verb. It is however difficult to quantify the extent to which the present state of the lexicon impedes the performance of the constraint relaxation technique compared to a 'perfect' lexicon.

Some error categories, such as auxiliary errors, would have deserved a better treatment, both on the computer science and on the linguistics points of view. We should be able to detect a passive structure from a erroneous use of the *être* (be) auxiliary. This error type being not very frequent, it was not considered a priority and time which would have been necessary for a full treatment was devoted to other error categories.

Another important limitation, which has been mentioned above in the section on problems encountered (page 130 and ff.), is the overgeneration of structures. In order to contain their growing number, we have been forced to relax some constraints more cautiously, such as the adverb order constraint. This prevents us from detecting some errors but reduces the number of spurious analyses and of overflagging occurrences to a more manageable level.

Informal assessment

While chapter 5 is devoted to tests, results, and evaluation of the whole diagnosis system, it might prove useful to provide here an informal assessment of constraint relaxation to conclude its implementation description.

Constraint relaxation is a useful technique for error diagnosis in good part because of the extent of error categories it can detect and diagnose. Constraints are linked quite straightforwardly to errors so that we can consider that a relaxed constraint indicates a specific type of error. However, to know exactly which constraints need to be relaxed, it is important to have a corpus with a large enough number of errors for each of the error subcategories one wants to detect. This was sometimes lacking in the present implementation[12] and we are not certain that all error categories and subcategories mentioned are covered in full. This could be amended at a later development stage of the diagnosis system.

As mentioned above, the main difficulty linked to constraint relaxation is the increased number of complete analyses provided by the system which causes problems in the selection process of the best alternative. One must present to the users the most likely analysis in terms of what they have written. This implies, for non-native writers, a certain likelihood for errors, while at the same time one must be careful to prevent occurrences of overflagging.

Nonetheless, twelve error categories[13] are detected through the constraint relaxation technique. This proves the interest for this diagnosis technique and the feasibility of its implementation in an existing syntactic parser.

4.3.2 Phonological reinterpretation

As we have seen in chapter 3, phonological reinterpretation has several prerequisites. These prerequisites are fortunately available with Fips and its family of applications. We recall them briefly here.

1. Fips can serve as a recognizer, although it tends to give full analyses for sentences which are not fully grammatical. Modifications made on the Fips parser in the context of constraint relaxation somewhat alleviate this problem by enforcing some new constraints and thus making the parser stricter.

2. It is one of the great advantages of Fips to provide partial analyses when a sentence is not fully analyzed. While useful in many contexts, this is a requirement for phonological reinterpretation in order to know which words to reinterpret.

3. Moreover, Fips is non deterministic, which facilitates the reinterpretation process as it allows all reinterpreted words to be tried at the same time.

4. Because of the speech synthesizer associated with Fips, FipsVox (Gaudinat and Goldman 1998), the lexicon used by Fips contains phonological information. Each word is associated with a phonological sequence, completed, if need be, by the information about latent consonants.

[12]Only extracts of the FRIDA corpus were made available to us.

[13]Consolidating the different order subcategories (ORDAV, ORDAJ) into a global ORD error category.

5. Routines have been set up to access the lexicon by the phonological information it contains. It is thus relatively easy to retrieve all the words with a given pronunciation.

Implementation of phonological reinterpretation was thus made possible by the presence of all the required tools. We did not need to create them. Our task consisted mainly in using them in appropriate ways to include this new layer of grammar error diagnosis within the parser to transform it into a diagnosis tool. In the remainder of this section, we first describe the variant used and detail the implementation steps. We then discuss the problems encountered and the extent and limitations of this implementation. We conclude with an informal assessment.

Variant employed

The implementation of phonological reinterpretation was done step by step and each main step is described in this section. Once the parse of the sentence has provided partial analyses, one must find the words at the chunk borders, determine their alternative pronunciations, find the corresponding list of reinterpreted words for each pronunciation, insert them in the chart used by the parser, analyze the sentence again, prepare the diagnosis, and customize the sentence for display.

Finding the words at the border of chunks This first step in the phonological reinterpretation process necessarily happens after the sentence has been through a first attempt at parsing which resulted in partial analyses or chunks. One must then retrieve the words at the border of chunks. This is not as simple as it may seem at first sight.

First of all, we are only interested in words which are at the border of two chunks, and not the first and last word of the sentence. If we consider the abstract representation of an analyzed sentence in (65), where each letter represents a word and each group of letters is a separate chunk, we should then consider only the words *c*, *d*, and *e* for phonological reinterpretation.

(65) *abc d efgh .*

One should not include *a* into the list of words to be reinterpreted, because it is the start of the of the sentence and it is normal that the chunk starts with it. The same applies in the reverse with *h*, although the case is slightly more complicated as sentences most often end with punctuation and this punctuation is automatically treated as a separate chunk by the Fips parser. One must also take care to include *d* only once in the list, and not twice, although it is the pending word of both *c* and *e*.

Moreover, one must take care to retrieve actual words and not a part of a syntactic structure containing an empty position. If the chunk under investigation contains, for example, a verb phrase inserted inside a tense phrase, but without a subject, one would like to retrieve the first overt element of the chunk, rather than the empty position. For the chunk shown in (66), it means that we recover the word *dort* (sleeps).

(66) $[_{\texttt{TP}}\ [_{\texttt{DP}}\ \text{e}\ ]\ [_{\overline{\texttt{T}}}\ \text{dort}\ [_{\texttt{VP}}\ ]\ ]\ ]$

Finally, at this stage of the phonological reinterpretation process, one is only looking for the alphabetical strings which represent the words, and not in the words' full lexical information as given by the partial analysis. Indeed, the full lexical information would restrict us to a given interpretation of the string, and thus to a given pronunciation, while we need to take into account all possible pronunciations for a given word string. The word *couvent* (convent / sit on (eggs) 3rd-plural-present) is pronounced /kuvã/ when it is a noun and /kuv/ when it is a verb. A chunk resulting from a partial analysis may provide the wrong interpretation for a particular word.

Thus, for each chunk of an incomplete analysis, we look at its starting and ending points. We access the graph containing all the analyses built so far by these starting and ending points and we take the word string we are interested in. The process is repeated for each chunk, with the restriction on the first and last chunks to prevent treating sentence front and end words, and the recovered strings are inserted in a list of words to be reinterpreted, along with the indication of their start and end positions in the sentence.

Determining all alternative pronunciations Once the list of all the word strings to be reinterpreted has been established, we must find all possible alternative pronunciations for each of these word strings. Each pronunciation is associated with a different lexeme of the given word string. In order to find the alternative pronunciations, one can thus look for the alternative lexical items of a given word string. A lexical item contains a great amount of information, such as the word string, its lexeme, its pronunciation, as well as many other syntactic and semantic features associated with it. The lexical items, with their complete information, are extracted from the lexicon which is accessed through the alphabetical information of the word string. The lexical items thus recovered, complete with pronunciation, are stored into another list.

Reinterpretation and insertion in the chart Provided with the list of alternative lexical items, and thus with the list of alternative pronunciations, it is now possible to reinterpret the words. Traversing the list of lexical items, one accesses the lexicon through the phonetic information in order to retrieve alternative lexical items with a corresponding pronunciation. One first takes the phonetic string as such, and, at a second stage, one concatenates it to its latent consonant, if one exists, in order to treat liaison phenomena which can be misinterpreted by learners as being part of the word. Each time a new lexical item is found, it is first compared to the original word string to make sure that one does not insert an already present lexical item in the graph, as this would only increase without reason the amount of ambiguity. Once this check is passed, the word is expanded into a projection. That is, it is inserted within a maximal projection of its own type. A noun is thus transformed into a noun phrase, a verb into a verb phrase. The projection is then inserted into the chart of constituents. If the projection can be further expanded into a larger constituent, this is done and the new constituent is also inserted in the chart. Thus, a tensed verb is inserted in the chart both as a verb phrase and as a tense phrase, as exemplified in (67) below.

(67)a. *mange* (eats)

b. $[_{\mathtt{VP}}$ *mange* $]$

c. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \mathrm{e}\ ]\ [_{\overline{\mathtt{T}}}$ *mange* $[_{\mathtt{VP}}\ ]\ ]\ ]$

Apart from considering individual words, it is also interesting to take into account the possibility of words which are split across a chunk border. In order for such split words not to be detected at the spell checking level, each part of the word must necessarily form a legal word of its own, as exemplified in the paragraph on multi-words in section 3.2.6. With each of the four possible combinations of the phonetic strings and latent consonants of two adjacent words at chunk borders, one reiterates the process of retrieving existing lexical items by a phonological access to the lexicon, of expanding the lexical items into projections, and inserting these projections into the graph.

Parsing the sentence again Once the new lexical alternatives have been inserted into the graph, it is time to re-parse the sentence. This step is not discussed further here as it pertains to the Fips parser itself. Suffice to say that the parser's non determinism makes it easy for it to try all the additional alternatives resulting from the phonological reinterpretation.

Fine-tuning the diagnosis There are several ways in which it is possible to fine-tune the diagnosis provided by phonological reinterpretation. Only one is currently implemented. It is to propose a diagnosis for reinterpreted words only when reinterpretation has allowed a sentence to be completely analyzed. The structure of incomplete analyses is less certain than that of full analyses. It is therefore more likely for a word to be considered in its wrong acceptation in terms of lexical category and/or homophone if it is an ambiguous word to start with. By rejecting partial analyses at this stage, one prevents a number of overdetection occurrences from cropping up.

Currently, all reinterpreted words receive a unique diagnosis which classifies them as HOM (homonymy) errors. However, some finer distinctions can be made for homophones of the same lexical category as the entry word. Methods to refine the diagnosis process are not implemented yet.

Customizing the sentence for display Phonological reinterpretation provides a correction of the sentence as the problematic words were replaced by better suited ones. Correction is not, however, what one is looking for in the CALL context. The mandate is to provide a diagnosis of the errors, in order to help the learners correct them by themselves. The output of phonological reinterpretation is thus not adapted. In preparing a reinterpreted sentence for display, one must insert the words of the original sentence back into the output, and one must provide a precise enough diagnosis.

When one inserts the original words back into the reinterpreted sentence, one does so only for those words which have been reinterpreted. The others have not been modified from the original, so there is no need to touch them at all. As for the words which must be transformed back, the whole lexical item has been modified during phonological reinterpretation, thus modifying either the lexical category of the words, or some other important syntactic features. While we want to display the sentence with its original words, we do not wish to replace the whole lexical item as this would most often be incompatible with the newly created sentence structure. Therefore, one changes only the word string of the reinterpreted word back to the original word string, leaving in the structure all the lexical information regarding the phonologically reinterpreted word. The sentence in (68a) is reinterpreted with the structure in (68b). Once the original incorrect word is reinserted into the structure, one obtains (68c) (which obviously would be displayed in a more user-friendly format in the software).

(68)a. **Elle à invité des gens.*
She to invited some people.

b. [$_{\text{TP}}$ [$_{\text{DP}}$ *Elle*] [$_{\overline{\text{T}}}$ *a* [$_{\text{VP}}$ *invité des gens*]]]
She has invited some people.

c. * [$_{\text{TP}}$ [$_{\text{DP}}$ *Elle*] [$_{\overline{\text{T}}}$ *à* [$_{\text{VP}}$ *invité des gens*]]]
She to invited some people.

The other important step in the output preparation is the insertion of the error mark, which is naturally linked to the precision of the diagnosis. So far, for all the reinterpreted words, whether they are actually homophones or not, the 'category' attribute of the 'error' tag of the XML display structure (described briefly in section 4.2.4) receives the value 'HOM' for homonymy. Finer distinctions should be made for words belonging to the same lexical category. Indeed, in such cases the error is not that of taking a homophone for another, but rather of using a word with the wrong set of syntactic and semantic features. Errors can be of gender, tense, mode, and so on, as it is the case for the second reinterpreted word in example (70), on page 143.

Problems encountered

Problems encountered with phonological reintepretation are of two kinds which are both linked to the increase in lexical alternatives, and which are thus unfortunately inherent to phonological reinterpretation.

As with constraint relaxation, the overgeneration of structures created during the parsing process for a given sentence is a major difficulty. When a sentence is proposed for reinterpretation, a minimum of two words will be reinterpreted, the maximum being the number of words in the sentence. The words which undergo phonological reinterpretation can introduce many alternatives into the chart. A word such as *est* (is/east) has at least the possible reinterpretations proposed in (69).

(69) *es*, *aie*, *aies*, *ait*, *aient*, *Est*

Therefore, from a word string with a double pronunciation and meaning, one gets a word string with six additional meanings, for a total of eight. This phenomenon being reproduced to some extent with each reinterpreted word, there being one or two reinterpreted word strings per chunk in a sentence, and often more than two chunks in an incomplete parse, one quickly reaches a combinatorial explosion in the number of alternatives to be tried during the parsing process. This combinatorial explosion leads in turn to an increase in the resources needed for parsing, in terms of time and memory space. The performance of the system thus suffers.

The increase in word ambiguity also leads to more complete analyses being found for a given sentence. Although it is the purpose of phonological reinterpretation to find a complete analysis for an ungrammatical sentence, exiting the process of phonological reinterpretation with several complete analyses is not ideal as it raises the question of which alternative structure to keep. As stated before, phonological reinterpretation is only invoked when a first attempt to parse a sentence completely has failed. One could thus legitimately hope that the number of alternative complete analyses with phonological reinterpretation would be relatively small. This is unfortunately not the case.

Multiple complete analyses are sometimes the result of several words being phonologically reinterpretated within the same sentence. While this is sometimes necessary, as in example (70), were the two words need reinterpretation to form a grammatical sentence, it can be detrimental if some of the reinterpretations are not necessary to build a complete analysis. The erroneous sentence in example (71a), can be corrected either as (71b) where only one word is reinterpreted, or as (71c) where two words are modified.

(70)a. **Il à aboutit.*
He to succeeds.

b. *Il a abouti.*
He has succeeded.

(71)a. **J'ai vu sait homme.*
I have seen knows man.

b. *J'ai vu cet homme.*
I have seen this man.

c. *J'ai vu sept hommes.*
I have seen seven men.

Obviously, only one of the analyses proposing the correction in (71b) and (71c) will be presented to the user, while both are possible. There is therefore a need to order and sort the complete analyses. This can be done by retaining the complete analysis with the fewest number of reinterpreted words, which would promote (71b). We thus systematically prefer a sentence where fewer errors are detected in order both to respect as much as possible what the user wrote and to keep the number of overflagging occurrences to a minimum. Moreover, in case of a tie, it might be more interesting to present the analysis with the highest number of true homophones, that is of words reinterpreted into a different lexical category, and not simply having different lexical or morpho-syntactic features. Errors indicating different feature sets are much more likely to be caught by other diagnosis techniques, such as constraint relaxation, than true homophones. Phonological reinterpretation should therefore concentrate on areas other techniques are not good at.

Extent and limitations of the implementation

The extent and limitations of the implementation of the phonological reinterpretation technique are described in this section.

Homophones and other erroneous words sharing a single pronunciation with the correct words can be detected, diagnosed and corrected within a sentence, as long as this sentence does not receive a complete analysis before phonological reinterpretation is activated and as long as the erroneous words are located at the border of the chunks returned as partial analyses.

Two interesting phenomena are so far treated in one direction only. They are latent consonants and multi-words. When a word is identified for reinterpretation, the list of the possible pronunciations of the word string is retrieved from the lexicon. The information on the pronunciation potentially contains a latent consonant. Thus, we reinterpret the word string with and without its latent consonant. This explains how the word in (72a) can be reinterpreted as words in (72b) and (72c).

(72)a. *sait* (pronounced /sɛ/ with /t/ as a latent consonant)
knows

b. *sais* (pronounced /sɛ/)
know

c. *cet*, *cette*, *set*, *sept* (pronounced /sɛt/)
this-masculine, this-feminine, set, seven

The reverse has not been implemented so far. It is therefore not yet possible to retrieve a word such as *sait* (/sɛ+t/)(knows) from the original *cet* (/sɛt/) (this). In order to accomplish this, we would need to modify the lexicon lookup procedure, and perhaps the indexing of the lexicon, to allow the lookup procedure to retrieve words on the basis of the combination of the standard pronunciation plus the latent consonant.

The reinterpretation process so far allows the combination of two adjacent words located at the border of chunks into one. This combination is computed on the pronunciation of the single words, with and without latent consonants. The resulting combined pronunciations can thus be reinterpreted as a single word. In many cases, these four pronunciations do not result in legal words and thus will not be inserted in the chart as new word alternatives to be taken into account. But from time to time, one (or a few) of the pronunciations can be reinterpreted as a legal word and is usefully added as an alternative.

(73)a. * *Vous pouvez peut être dormir.*
You can-2nd-plural can-3rd-singular be sleep.

b. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{vous}\ ]_i\ [_{\overline{\mathtt{T}}}\ \text{pouvez}\ [_{\mathtt{VP}}\ ]\ ]\ ]$
$[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{e}\ ]\ [_{\overline{\mathtt{T}}}\ \text{peut}\ [_{\mathtt{VP}}\ [_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{e}\ ]\ [_{\overline{\mathtt{T}}}\ [_{\mathtt{VP}}\ \text{être}\ ]\ ]\ ]\ ]\ ]\ ]$
$[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{e}\ ]\ [_{\overline{\mathtt{T}}}\ [_{\mathtt{VP}}\ \text{dormir}\ ]\ ]\ ]$

c. *Vous pouvez peut-être dormir.*
You can perhaps sleep.

d. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{vous}\ ]_i\ [_{\overline{\mathtt{T}}}\ \text{pouvez}\ [_{\mathtt{VP}}\ [_{\mathtt{AdvP}}\ \text{peut-être}\ ]\ [_{\overline{\mathtt{V}}}\ [_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \text{e}_i\ ]\ [_{\overline{\mathtt{T}}}$
$[_{\mathtt{VP}}\ \text{dormir}\ ]\ ]\ ]\ ]\ ]\ ]\ ]$

This enables us to at least recuperate the words *peut être* (can be) in sentence (73a), which gives the partial analyses in (73b), to interpret them as only one word, *peut-être* (perhaps), which permits in turn a complete analysis of the sentence as shown in (73d).

We can therefore currently reinterpret a word split into two words in the input into a single word. However, we are not yet able to do the reverse, that is reinterpret a single word in the input as two words. In order to do so, one would need to be able to access the lexicon phoneme by phoneme in order to find all the possible words contained in the input word pronunciation. Such an access to the lexicon is not currently available to us and phonological reinterpretation finds itself therefore limited in its current implementation.

At the present time, words can be retrieved through phonological reinterpretation only if their pronunciations are identical to that of the original words. No approximation is currently possible. The inclusion of approximation within phonological reinterpretation would however be greatly facilitated by existing routines which, from a given phonetic string, compute all possible phonetic strings which vary only in vowel aperture. Obviously, implementing this option would also increase the number of alternatives to be treated by the parser, with the damaging effect this is bound to have on performance. Parameterizing which sounds can be modified on the basis of the users' mother tongue would be a very interesting option. We do not have, however, the corpus information which would allow us to determine groups of variable sounds for native speakers of given languages.

Precise diagnosis information is not available at this stage either. Every word which is reinterpreted receives an HOM error tag, without close comparison with the original word. There is, therefore, no distinction between true homophones (of different lexical categories) and different forms of the same root word, such as *aboutit*/*abouti* (succeeds/succeeded), in example (70) or *homme*/*hommes* (man/men), in example (71). This does not prevent phonological reinterpretation to detect and provide a good correction for these errors, but the diagnosis component, the precise identification of the error type, suffers from it.

Informal assessment

Although formal testing of phonological reinterpretation is discussed in chapter 5, let us briefly make an informal assessment of it here. Phonological reinterpretation appears to work rather well in terms of the errors it is capable of detecting and correcting. In terms of diagnosis, as stated above, we are currently limited to only one denomination of the errors found, namely that they are homophones.

In the chosen variant of the phonological reinterpretation, very complex sentences containing several errors tend not to receive a diagnosis through phonological reinterpretation. This is due to the decision not to show the users the results of phonological reinterpretation if this technique was not able to provide a full analysis of the sentence, in order to reduce the number of overflagging occurrences. Long, complex sentences with several errors have a higher likelihood of containing errors not diagnosed by phonological reinterpretation and, thus, of not reaching a full analysis.

The main difficulty with phonological reinterpretation is the great number of alternatives which need to be tried as it is only by trying the reinterpreted words that one can know whether they are valid leads for correction and hence diagnosis. This potentially great number of additional alternatives, combined with the need to parse the sentence again, makes the process slower.

4.3.3 Chunk reinterpretation

Chunk reinterpretation has not been implemented at the present time and we do not foresee doing it in the near future. Reasons for not implementing the chunk reinterpretation technique are explained in this section.

The theoretical description of chunk reinterpretation which can be found in chapter 3, section 3.3, is far from being precise enough to count as valid specifications. The first step in implementing chunk reinterpretation would have been to write very precise specifications which would have been readily implementable. In order to do so, one would have had to study a large amount of data in the form of parsed sentences with incomplete analyses. Although these data are not difficult to create, it is more delicate to predict how large an amount would have been necessary in order to be able to categorize and sort the partial analyses into different groups linked to distinct error categories. No doubt the process of designing the specifications would have been time consuming. Moreover, there might be differences in the type of information one would have been able to extract from the data, depending on whether other diagnosis techniques were applied to the sentences before the parser returned chunks or not. Thus, an *a priori* decision on the type of technique combination would have been judicious in order to get the most of the chunk reinterpretation technique in the precise context it would have been employed in.

The second important point which would have needed to be taken into account is the problem of reliability of the technique. Based on partial analyses, like phonological reinterpretation, chunk reinterpretation needs the chunks returned by an incomplete parse to be the most appropriate ones, that is to break at the error location exactly. This is however not guaranteed by the chunk selection process in Fips which tries uppermost to find a combination of the longest chunks available. The error location might not be at the chunk boundaries as often as one would like them to be. Besides, chunk reinterpretation itself depends vastly on a thorough study of the data provided by partial analyses in order to derive the reinterpretation rules. Rules retained will be those which cover the greatest number of cases with the fewest false positives. In a number of ways, this can be assimilated to heuristics, or *ad hoc* measures, rather than a really scientific, proven diagnosis technique. One can then wonder how reliable such a technique would have been in practice, but obviously only its implementation and testing would have been able to give us trusty information on that score.

The third problematic issue which did not particularly incite us to implement chunk reinterpretation is its lack of portability to other systems. As we have stated above, the rules designed during the specification stage are necessarily extracted from a close analysis of chunks output by the parser. This close dependency between the rules and the output of the parser bodes poorly for portability. It is rather likely for another parser to react differently to the input, to use a slightly different version of the linguistic theory behind it, or to have another chunk selection process. Thus, the whole set of rules would have to be completely redesigned on the basis of a new set of data provided by the parser one wants to adapt the technique for. This would basically amount to redoing the whole process and is a strong indication of the heuristic or *ad hoc* nature of the chunk reinterpretation technique, as it was mentioned in the preceding paragraph.

The last factor which prevented the implementation of chunk reinterpretation is the less satisfactorily but perhaps also the most concrete one: lack of time. This lack of time showed itself in two dimensions which are the project in which the error diagnosis system is implemented, and the dissertation work itself. From the project stand point, implementation of constraint relaxation and phonological reinterpretation took longer than expected, in good part because we were faced with an enormous level of structure overgeneration which we had not imagined and for which we had to devise renewed alternative selection procedures, as described in section 4.3.4 below. This left no time for the implementation of chunk reinterpretation within the project's time frame. On the dissertation front, the need to progress in the written part of the work in order to meet approaching deadlines necessarily lead to some choices as to where efforts should be best spent and it seemed that it would be more appropriate to provide comprehensive results of the two implemented diagnosis techniques than to implement a third one.

However, we can note by looking at Table 4.1 (page 122), that all the error categories covered by chunk reinterpretation are also covered by either or both of the other diagnosis techniques, namely constraint relaxation and phonological reinterpretation. This redundancy implies that, by not implementing the chunk reinterpretation technique, we should not unduly damage the total number of error categories the whole diagnosis system is capable of treating.

Therefore, for the reasons mentioned above, some better than others one must admit, the implementation of the chunk reinterpretation diagnosis technique has not yet been realized. This does not imply that the technique is worthless in itself, it is rather a shift in priorities. The ideas behind chunk reinterpretation are not abandoned and one might see their implementation outside the scope of this dissertation work.

4.3.4 Analyses selection mechanism

In an attempt to find the best way to automatically select the best analysis alternative from those proposed by the error diagnosis system with constraint relaxation and phonological reinterpretation, three scoring mechanisms were implemented to complement the original scoring mechanism available in Fips. Different ways to combine these scores were tried out and tested on a parsed corpus. We first describe the three additional mechanisms, which differ somewhat from their original description in section 3.4. We then explain the tests which were conducted and expose the solution we have currently adopted in view of the results.

Scoring mechanisms

The different paragraphs below mirror those from chapter 3, section 3.4, and relate the differences between the theory and the actual implementation in the diagnosis system which serves as a basis for this dissertation.

Syntactic likelihood Two scoring mechanisms are used to measure syntactic likelihood. The first one is the score embedded within Fips and which was designed for grammatical sentences. The score is computed by adding and subtracting points at many levels in the syntactic analysis. The number of points added or subtracted depends on many factors, including the words themselves, the attachment points, and some further lexical and syntactic features. Ties between alternative analyses are allowed and are frequent.

In section 3.4.1, we mentioned using bi- or n-gram frequencies in order to sort the alternatives. This path was not pursued further, however, due to the lack of an appropriate corpus from which to derive the relevant statistics. On the other hand, a score computing the number of nodes of a structure, the maximal depth of the structure, and the number of nodes at the maximal depth was implemented. Calculating this metric is done after parsing is completed, which has the distinct advantage of not interfering with other components of the error diagnosis system.

Error frequencies The formula proposed for computing error frequencies in section 3.4.2 proved more complex than necessary and was thus simplified in the implemented error frequency scoring mechanism. First of all, normalization of the score by the length of the sentence is not useful in the present case, as one tries to sort alternative analyses of the same sentence. Thus, all the sentences under scrutiny at one point in time share the same number of words. Normalizing by the length of the sentence would not be discriminative. Secondly, it seemed that multiplying the score by a factor taking into account the number of errors detected in the sentence was redundant with the addition of the relative frequencies of each of the errors found in the sentence analysis. Finally, instead of using the modified reverse percentage proposed initially, we used the logarithm of the error frequency, with up to four decimal points. This gives us low numbers for frequent errors and higher ones for less frequent errors. The error frequencies are based on the statistics of the FRIDA corpus (section 4.2.1) and estimated for error categories not included in FRIDA (NEG) or for which we treat only a part of the FRIDA category (CLA and OUB, for example). Thus, the formula for the error frequency scoring mechanism became:

$$\sum_{i=1}^{|e|} \log\left(f\left(e_i\right)\right)$$

Where $|e|$ is the number of errors in the sentence and f a function returning the frequency, up to four decimal points, of the error in the FRIDA corpus.

With this scoring mechanism, we still promote error-free analyses, which receive a score of 0, and frequent error categories over less frequent ones.

Word frequencies Word and lexeme frequencies are part of the information contained in the lexicon used by our error diagnosis system. Although the corpus which was used to retrieve those frequencies is of journalistic origins and not necessarily the most appropriate in the CALL context, we decided to make use of these data instead of creating new ones, for simplicity's sake. Both word and lexeme frequencies are combined in the metrics we use. Once the parsing phase is achieved and all the alternative analyses have been collected, one computes the word frequency score of each alternative. This implies accessing each word in turn in order to add its word frequency to the score being computed. However, if a word can have several lexemes, the word frequency is not directly added to the score, but it is first pondered by the lexeme frequency of the lexeme used in the particular case.

Tests on a corpus

In order to test the efficiency of the four different scoring mechanisms implemented (Fips original score, node complexity, error frequency, and word frequency), we required some data. We parsed, with the error diagnosis system, a 10'000 word extract from FRIDA and, for each of the roughly 500 sentences, the best alternative analysis was manually selected. We thus obtained a parsed corpus from which statistical data could be extracted on the single and combined use of the different scoring mechanisms.

Two different series of tests were run on the parsed corpus. In both cases, the manually selected analysis was compared in turn with each non-preferred analysis of the same sentence in order to derive the fitness of each scoring technique. The first series of test employed statistics, while the second made use of fuzzy inferences. Both tests were designed and conducted by a programmer with good knowledge of statistics and fuzzy logics. The tests, however turned out to be disappointing in that they were not able to establish an optimal way in which to weigh and combine the different scoring mechanisms in order to retrieve the best analysis in a reliable manner. Several reasons for this lack of usable results were invoked, namely too much overgeneration in the input data, inadequate scoring methods, and lack of information on the rejected analyses. This lead us to redesign, at least on a temporary basis, the selection mechanism for alternative analyses.

Current selection mechanism

We were not able to find a satisfactory way to combine the different scoring mechanisms in order to obtain the best alternative analysis most of the time, but we needed at one point to freeze the development of the system in order to be able to run some tests on a stable version. General results showed that the original Fips scoring mechanism was still the one which worked best overall. It thus seemed important to take this score strongly into account. At the same time, we wanted our error diagnosis system to have as few occurrences of overflagging as possible, while still indicating error diagnoses for erroneous sentences.

In order to minimize overflagging, we decided to separate the alternative analyses into two parts, the first one containing error-free alternatives, and the second error-full ones. In the normal case, we present the error-free analysis with the lowest original Fips score to the user. For example, an error-free analysis with a score of 75 will then be promoted over an error-full analysis with a score of 50. This is however tuned down by also looking at the scores of error-full analyses. If an error-full analysis has an original score of less than half the score of the first error-free analysis, then this error-full analysis is displayed instead of the error-free one. Thus, an error-full analysis with a score of 30 would win over an error-free analysis with a score of 75. Moreover, in case it is decided to show an error-full sentence analysis, all the alternative analyses which contain errors are ordered according to the error score. This is another attempt to avoid overflagging as much as possible.

4.3.5 Choice of technique combination

Constraint relaxation and phonological reinterpretation are the two diagnosis techniques which have been implemented and which need to be combined in order to provide a coherent diagnosis.

The sets of error categories which constraint relaxation and phonological reinterpretation are able to detect share some of the same categories but are mostly distinct. In fact, only agreement errors overlap in any significant way. Moreover, phonological reinterpretation is able to detect some errors of agreement only, provided the agreement features are not overtly realized, while constraint relaxation treats all types of agreement errors in gender, number and person. For these particular error categories, constraint relaxation is thus more adequate. This should not surprise anyone, as it is well attested in literature that "relaxing constraints of an existing grammar is particularly suitable for identifying agreement errors" (Tschichold 1999, p. 210).

It thus seemed possible to decide that agreement errors would be marked only if they were detected by constraint relaxation, *de facto* separating the sets of error categories detected by each diagnosis technique. The independent approach to technique combination described in section 3.5.1 seemed therefore easier to implement, as there would be no risk of error diagnosis overlap. However, phonological reinterpretation requires a first pass through the system in order to obtain chunks. Although we had designed our diagnosis system to have a first pass without any diagnosis technique activated in order to filter out grammatical sentences, this first pass was never implemented.[14] Skipping this first pass has the advantage of shortening processing time for ungrammatical sentences. However, it prevents using phonological reinterpretation before, or in parallel with, constraint relaxation. Thus, the independent approach is not possible.

Both diagnosis techniques require a rather large amount of resources to run, both in terms of memory and time, as they often overgenerate quite widely. Moreover, as it is rather unlikely to find all types of errors in a single sentence, we decided to use the cascade approach (see section 3.5.2) and to launch the second technique only if the first one failed to give satisfactory results. Because (i) constraint relaxation is able to diagnose a wider range of errors than phonological reinterpretation, (ii) the errors of this range tend to occur more frequently than those diagnosed by phonological reinterpretation, and (iii) phonological reinterpretation requires partial analyses, it was decided that constraint relaxation would be activated first.[15] As phonological reinterpretation needs partial analyses as ground material, the technique is activated only when no full diagnosis can be reached even with relaxed constraints. Thus, phonological reinterpretation is called only when constraint relaxation is not sufficient to diagnose all errors present in

[14]Despite several requests to the main programmer in charge of constraint relaxation to do so.

[15]Note also that this corresponded to the implementation order of these two diagnosis techniques.

a sentence.

We pondered the possibility of activating phonological reinterpretation together with constraint relaxation. That is, we would have a first pass with constraint relaxation activated only, and, if need be, a second pass with phonological reinterpretation and constraint relaxation activated together. This option was rejected on the ground of generating too many false positives.

(74)a. **set animaux*
set animals

b. *sept animaux*
seven animals
Homonymy error on the determiner

c. **cet animaux*
this animals
Homonymy error on the determiner
Number error between the determiner and the noun.

Example (74a) above can be reinterpreted into (74b) quite straightforwardly, with or without constraint relaxation being activated at the same time. On the other hand, (74c) is a possible correction only if constraint relaxation is activated at the same time as phonological reinterpretation. It is however not very likely to have a number error on *animaux* (animals) in this particular case and (74b) is much more likely the best correction. Depending on how the alternative analysis selection works, there is a risk of selecting (74c) as the best analysis, instead of (74b), if constraint relaxation is allowed while the phonological process is underway, and thus of displaying an overflagging occurrence (the number error) to the users. Therefore, the decision was taken not to allow phonological reinterpretation and constraint relaxation to work at the same time.

4.4 Conclusion

This chapter has presented the implementation phase of two of the error diagnosis techniques described theoretically in chapter 3 of this dissertation. In order to expose the precise implementation situation, we first described the Fips syntactic parser which serves as a basis for our diagnosis system. We then discussed the error categories that we intended our diagnosis system to be able to diagnose, based on corpora studies and on the theoretical possibilities of the techniques we were implementing. Fourteen error categories were finally selected and briefly described. We went on to describe the actual implementation of constraint relaxation, relating the precise variant we used, the kinds of problems we encountered, as well as the extent and limits of the current implementation. We then turned to the implementation of phonological reinterpretation, here again indicating which version of the technique was used, the problems which we were faced with, as well as the extent and limits of the implementation. We briefly explained the reasons which led us to forego the implementation of chunk reinterpretation. We then focused on the analysis selection mechanisms and the choice of technique combination.

The major difficulties which were encountered in the implementation phase of the error diagnosis system were the overgeneration of alternative analyses, which we had underestimated. This overgeneration is due both to constraint relaxation and to phonological reinterpretation. In turn, it also leads to a potential problem of error overflagging as many of the alternative analyses, in order to exist, have to make use of constraint relaxation or phonological reinterpretation. Use of either technique implies the insertion of error marks in the structure, which are displayed to the user if they are part of the alternative analysis selected as the most likely one.

Time issues prevented us from playing around as much as we wanted to with different parameters in order to try to discover the best settings to maximize the results obtained by the error diagnosis system. In particular, it would have been interesting to modify the number of passes into the parser, with different packets of constraint relaxed at different times. One can easily imagine that many permutations are possible, each requiring exhaustive testing in order to be able to measure one solution against another. Effectively trying different technique combinations, including different ways to handle converging and diverging results from the two implemented techniques, would have been interesting as well, although very time consuming.

Even implementation phases, which could go on for ever with testing, improvements and bug fixing working in a never ending loop, must end at one point. We thus froze the development of the error diagnosis system at a given point in time. It is the resulting version of the error diagnosis system which is presented in this dissertation. This gave us the stable version we needed in order to run tests on the system, in order to verify whether the error diagnosis system which was built was usable in the context of a CALL system used by intermediate to advanced adult language learners. Chapter 5 thus explains the tests which were conducted in order to determine how good, or bad, our system is in terms of diagnosed errors, of undetected errors, and of overflagging occurrences.

Chapter 5

Results

After a theoretical description of error diagnosis techniques (chapter 3) and a report on their actual implementation (chapter 4), it is now time to discuss the results obtained through the use of these error diagnosis techniques. This is a way to control whether the theory keeps its promises and whether the techniques are actually usable in a wide scope error diagnosis system such as the one described in this dissertation work.

An evaluation of the prototype needs to be performed along at least the three following axes:

1. quantitative results, i.e. actual numbers of errors detected or undetected in specific corpora;

2. qualitative results, i.e. a more subjective general appreciation of the system;

3. efficiency of the system, i.e. an estimate as to whether the resources needed by the system stay within acceptable limits.

All of them are important to form a complete picture of the error diagnosis system as it now stands. It is only by having a clearer image of the system in action that we can try to draw conclusions as to its interest for actual language learners as a tool, integrated or not within a larger CALL system.

In this chapter, we start by describing the types of corpora which have been used to test the error diagnosis system (section 5.1). We continue in section 5.2 by detailing the different results obtained on different corpora and for the different error types which are supposedly treated by our error diagnosis system. We then compare our system to a commercial system in section 5.3, before concluding the chapter (section 5.4).

5.1 Description of the test corpora

To test the error diagnosis system (EDS, for short) and its implemented error diagnosis techniques, we have been using three different types of corpora. Each of them has distinct advantages over the others. Two types of error-full corpora were tested for six error categories, but only one for the remaining categories. Combinations of tests with different types of error corpora help us to form an overall picture of the diagnosis system.

In the following sections, we describe the three types of corpora, starting with the most artificial one (5.1.1), moving to a corpus closer to reality (5.1.2), to end up with almost authentic learner productions (5.1.3).

5.1.1 'Linguist' sentences

The first type of corpus is composed of 'linguist' sentences. By this term, we understand sentences which have been thought of and created only with the goal of building such a corpus. The main characteristics of such a corpus are

- relatively short sentences (sometimes even only parts of sentences);
- controlled vocabulary items;
- only one error per sentence;
- specific sentence structures to test particular configurations of the error category.

A corpus of 'linguist' sentences is an interesting device to test that the targeted error category is well detected in the expected syntactic configurations. It is easy to modify one or two parameters of the sentence, such as a vocabulary item, in order to verify possible reasons for undetected errors or overdetected ones. This type of corpus is thus especially helpful in the first stages of development in order to test the system while still being able modify it according to the test results.

For a final evaluation stage, 'linguist' sentences are useful because they might cover some areas of the error category under examination which are not part of more authentic corpora because of a sparse data problem. In particular, when an error subtype is especially rare in the authentic data, it might still be important to have several examples of it for testing purposes. However, one must note that 'linguist' sentences are usually relatively shorter and simpler than sentences in more authentic corpora, and that there may well be sentence structures never attested in actual language learner productions.

A sample corpus made up of 'linguist' sentences for the verb complementation (CPV) error category is provided in appendix B.1.

5.1.2 Simplified sentences

A more authentic corpus is composed of what we call 'simplified' sentences. The source of this corpus is authentic sentences produced by language learners and extracted from the FRIDA corpus (for a description of FRIDA, see section 4.2.1). These sentences have then been simplified by isolating only the clause in which an error of the relevant category occurred, by correcting any spelling mistake and all other errors in the sentence. In so doing, one creates a corpus perfectly targeting one error category while still taking into account sentence structures used by language learners and the specific errors they produce.

The main characteristics of a corpus composed of simplified sentences are

- actual errors made by language learners are present;
- there can be several errors in a sentence, but all should be of the same error category;
- sentence structures used by language learners are used, as long as they are grammatical;
- longer sentences are shortened to focus on the error category.

Thus, this type of corpus is already closer to what learners are going to be inputting into the error diagnosis system. Obviously, however, only corpus extracts are taken into account, and one can therefore easily miss some error-full constructions produced by language learners but not included in the extracted sentences.

A sample corpus of simplified sentences for the CPV error category is presented in appendix B.2.

5.1.3 Authentic sentences

The last corpus type is composed of nearly authentic sentences. By this, we understand sentences which have been actually produced by language learners, without being simplified. The only modification we permitted ourselves on these sentences was to correct spelling mistakes with the help of a spell checker. We tried to be loyal to the learners' intentions by substituting erroneous words by what seemed the most likely proposal of the spell checker and trying to imagine what the learners would have chosen, had they been using a spell checker. Spelling corrections were deemed appropriate in view of the fact that EDS does not include a spell checker but that it is supposed that one is used before the error diagnosis system is activated.

The main particularities of this corpus type are:

- several errors of different error categories can be found in the same sentence;
- apart from spelling errors, language learners wrote the sentences as such;
- sentences tend to be longer, more complex and more error-full than in the other corpus types;
- sentences are as close as possible to the predicted input of the target user population.

Moreover, corpora composed of authentic sentences used for this evaluation can be further subdivided into two subtypes depending on the origin of the sentences. Our main source of authentic data, for the development and implementation phases as well as for the evaluation phase, has been the FRIDA corpus. The FRIDA corpus, as described in chapter 4, section 4.2.1, is essentially composed of hand-written essays produced by language learners and then keyed in by the CECL team. A sample corpus is available in appendix B.3.1.

A second source of data comes from a CALL software prototype tested with a student population within the FreeText project (a brief description of the project was given in section 1.4). Sentences from this second source were typed in directly by the learners in answer to specific questions or exercises encountered within the FreeText software, that is within the kind of application in mind when designing this error diagnosis system. Unfortunately, this second source of data, which we will call 'FreeText data' for convenience sake, is not very rich in terms of sheer quantity of data. Moreover, the FreeText data was available quite late in the development of the error diagnosis system and had not, at the time of the experiment, been error tagged in the way the FRIDA corpus has been. Therefore, using it implied first marking down all the errors that are found in it, and especially the errors which EDS should be able to detect, in order to be able to verify the actual performances of the diagnosis techniques. The whole corpus of authentic sentences we used from this source is available in appendix B.3.2.

Differences between these two corpus subtypes, apart from size and error tagging, are linked to the original writing medium. The FRIDA sentences were first hand-written and then transcribed faithfully. However, it sometimes happens that words in the original paper corpus were unreadable and were replaced with question marks to indicate that a word was not recovered. It makes the automatic treatment of these sentences very difficult. The FreeText data, on the other hand, is more likely to contain typographical errors and to miss a number of diacritics as it was directly typed by the learners.[1] The FreeText data is thus more adequate overall, but its main

[1] Missing diacritics are especially frequent in typed data from learners whose first lan-

drawbacks, small size and lack of error tagging, makes it unsuitable for large amounts of testing.

5.1.4 Use of the different corpora

The diagnosis of the error categories were tested individually, using the three types of corpora in different ways. Most categories were treated with simplified sentences. Some used 'linguist' sentences as well, essentially when there were not enough authentic examples to make a real test corpus (class errors, for instance), or when precise knowledge of the information contained in the lexicon proved useful to know whether the technique was adequate or not (as in the case of verb complementation errors). Authentic sentences (FRIDA data) were used with caution as it was difficult at times to decide whether error detection and diagnosis in the specific error category under examination were perturbed by other errors within the sentences. FreeText data was used for the comparison with a commercial grammar checker.

The same corpora were used at different stage of the design and the implementation of the error diagnosis techniques and one might argue that the actual implementation of the techniques was tailored to the specific data at hand, thus providing better results than on random data. This might be partially true, but one must deny having knowingly tried to achieve this effect. Reusing the same corpora happened because gathering error specific set of data is time consuming and involved, for simplified and authentic sentences, relying on the CECL team who efficiently provided us with the corpora extracts, but whom we could not ask to repeat the task several times. The FreeText data, however, was not used at all during the development and implementation phases and thus can act as a neutral corpus.

5.2 Evaluation

An evaluation of the error diagnosis system as such, independently from other systems, has been conducted. The version of the error diagnosis system used for these tests includes both the constraint relaxation technique and the phonological reinterpretation technique, but not the chunk reinterpretation technique as it has not been implemented so far. The two used techniques are combined following the cascade method, phonological reinterpretation being activated only when constraint relaxation has not been able to provide a full analysis for a sentence. Moreover, when phonological reinterpretation is activated, constraint relaxation is disabled in order to reduce the number of alternatives to be tried out by the system and thus to reduce the processing time. To keep overflagging to a minimum as well, one displays errors found

guage uses basically no diacritic and whose keyboards do not make typing diacritics an easy matter.

by phonological reinterpretation only when the technique has permitted the system to reach a full analysis for the sentence at hand.

In this section, we first look at quantitative results for each of the error categories treated by the error diagnosis system (5.2.1). We used corpora of the different types described in section 5.1. Some qualitative results follow (5.2.2), based on less systematic tests, but more on a general impression of the error diagnosis system and of its diagnosis techniques. We then discuss the efficiency of the system as a whole (5.2.3).

5.2.1 Quantitative results

The error diagnosis system was tested on fourteen error categories. For each of these categories, test files of different types of corpora were set up and processed by EDS. A precise count of the number of errors in the corpora, of detected and diagnosed errors, and of undetected errors was made in each case. These results are reported here, hopefully in a sufficiently similar format for each category to allow for easy comparison. Complete similitude between the error categories is however impossible as the types of testing corpora used are not always the same, nor the number of sentences in them.

In each case, we distinguish between the number of errors actually present in the corpus, the number of errors which have been correctly detected and diagnosed, errors which have been detected but which have received an incorrect diagnosis tag, as well as overdetected or overflagged errors.

It is easy to compute the error recall rate[2] for all the types of corpora. However, a precision rate[3] is not informative on corpora composed of either linguist or simplified sentences as the sentences in both types of corpora are biased by removing all difficulties besides one specific error per sentence. Thus a recall rate will be provided systematically, but a precision rate will be stated only when discussing an authentic corpus.

Auxiliary errors (AUX)

Errors of the auxiliary category are relatively few in number. The error diagnosis system was tested on this error category using three different corpora. The first one is made up of linguist sentences, each containing an auxiliary error. The second corpus mirrors the first one, but all the errors have been corrected. Finally, the third corpus contains simplified sentences.

Table 5.1 shows that only a third of the auxiliary errors of this corpus are correctly diagnosed. However, another third of the errors are detected

[2]The recall rate is the percentage of errors in the corpus actually diagnosed and/or detected by the error diagnosis system.

[3]The precision rate is the number of diagnosed errors actually present in the corpus compared to the total number of errors detected by the system, including occurrences of overflagging.

Table 5.1: Linguist sentences with auxiliary errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	12	4	4	4	0
Percent	100	33.3	33.3	33.3	-

although given misleading diagnosis labels: two errors are considered to be voice errors (as in example (75)) and the other two homonymy errors (as in example (76)). Recall for detected errors, thus combining both correctly diagnosed errors and simply detected ones, reaches 66.7%.

(75) **Tu t'as évanoui.*
You yourself have fainted.
Voice error on *évanoui* (fainted).

(76) **Il est dormi.*
He is slept.
Homonymy error on *est* (is). Proposed correction: *ait* (have-subjunctive).

The second corpus, containing corrected versions of the sentences in the first corpus, did not present any auxiliary error. However, three errors were detected with a voice error tag.

Table 5.2: Simplified sentences with auxiliary errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	26	8	0	18	0
Percent	100	30.8	0	69.2	-

Results of the third corpus, made of simplified sentences, are summarized in Table 5.2. Only eight of the twenty-six errors were diagnosed, giving a recall rate of 30.8%. There were no other detection of errors on the precise location of the auxiliary error. For a number of undetected errors, the problem seems to be linked to an incorrect analysis of the structure of the sentence, and/or to incorrectly disambiguated lexical items. But these cover only seven sentences, leaving eleven unaccounted for.

One should also note that each time an auxiliary error has been detected, it is always the *avoir* (have) auxiliary which has been employed instead of the expected and grammatical *être* (be) auxiliary. We are thus able to detect errors such as in (77).[4]

[4]Note also that when an incorrect auxiliary is used, no check is performed on the agreement of the past participle. Past participle agreement rules in French strongly depend

(77) **La pillule* ***a*** *apparu pendant les années 60.*
The pill has appeared during the years 60's.

Recall for auxiliary errors is low with linguist sentences and even lower with simplified ones. Results are therefore not very satisfactory.[5] On the plus side, however, there is no real occurrence of overflagging and thus users of the error diagnosis system can trust the system when it indicates an auxiliary error. They must nonetheless be aware that probably not all of their auxiliary errors are detected.

Class errors (CLA)

Errors of the class category treated by our error diagnosis system are very limited. This category was tested only on the incorrect use of the *ce* (this) pronoun as subject of a verb other that *être* (be).[6] For this error category, we were not able to provide a corpus of simplified sentences, as there were too few examples, but we were forced to work with a very small set of linguist sentences only. Test results are summarized in Table 5.3.

Table 5.3: Linguist sentences with class errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	4	2	1	1	0
Percent	100	50	25	25	-

The CLA errors in two sentences were not shown by the error diagnosis system because the first analysis provided by the system was not the expected one. In both cases, a further analysis gave the appropriate result. One of these sentences is presented in (78), with the erroneous sentence in (78a), the first but incorrect analysis in (78b), and finally the correct analysis, complete with error tag, in (78c).

(78)a. **Ce montre qu'il l'avait vu.*
This watch/shows which/that he it had seen.

on the auxiliary used. When an auxiliary is incorrectly used, we do not know whether to apply the agreement rule with the current or the correct auxiliary. Thus we prefer doing nothing. A second run through the diagnosis system, once the auxiliary error has been corrected, would then detect agreement errors, if there are any.

[5]Remember the difficulties encountered during the implementation phase for the treatment of the erroneous use of the *être* (be) auxiliary and described in the section entitled "Extent and limitations of the implementation", page 132.

[6]Many class errors (as defined by the CECL), such as the choice between a definite or indefinite determiner, are too semantic in nature to be treated by a syntactic grammar checker.

b. $[_{\text{DP}}$ *ce* $[_{\text{NP}}$ *montre* $[_{\text{CP}}$ $[_{\text{DP}}$ *qu'* $]_i$ $[_{\bar{\text{C}}}$ $[_{\text{TP}}$ $[_{\text{DP}}$ *il* $]$ $[_{\bar{\text{T}}}$ *l'*$_k$ *avait* $[_{\text{VP}}$ *vu* $[_{\text{DP}}$ e$_k$ $]$ $[_{\text{DP}}$ e$_i$ $]$ $]$ $]$ $]$ $]$ $]$ $]_i$ $]$
This watch which he it had seen
Gender error between *ce* (this) and *montre* (watch).

c. $[_{\text{TP}}$ $[_{\text{DP}}$ *ce* $]$ $[_{\bar{\text{T}}}$ *montre* $[_{\text{VP}}$ $[_{\text{CP}}$ *qu'* $[_{\text{TP}}$ $[_{\text{DP}}$ *il* $]$ $[_{\bar{\text{T}}}$ *l'*$_k$ *avait* $[_{\text{VP}}$ *vu* $[_{\text{DP}}$ e$_k$ $]$ $]$ $]$ $]$ $]$ $]$ $]$ $]$
This shows that he it had seen
Class error between *ce* (it) and *montre* (show).

In both analyses, there is an error: a gender error between *ce* (this-masculine) and *montre* (watch-feminine) in the first analysis (78b); the expected class error between *ce* (this) and *montre* (show) in the second (78c). Thus the problem does not lie here in the detection of the error, but rather in the selection of the appropriate alternative analysis provided by the system.

The number of sentences is really to small to speak of recall or precision rates. We can note, however, that in a second corpus containing seven grammatical sentences with the kind of structures encountered in the corpus of erroneous linguist sentences, no class error was detected. This result is probably very dependent on the tight restrictions we imposed on errors of this category, but it is nonetheless encouraging.

Adjective complementation errors (CPA)

Errors of adjective complementation are already more frequent than auxiliary errors or our limited version of class errors. Choosing the appropriate preposition, whether it follows an adjective or a verb, is a particularly difficult task for language learners. We have been testing this error category with a corpus of simplified sentences only, without resorting to made-up sentences. The test corpus contains more sentences, which are longer and more diversified in kind. This test corpus contains sentences which have been less simplified than in many other cases. Test results are summarized in Table 5.4.

Table 5.4: Simplified sentences with adjective complementation errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	40	16	0	24	1
Percent	100	40	0	60	-

Recall on adjective complementation errors is only of 40%, which is not altogether impressive, although already better than for auxiliary errors. We are able to detect errors on both nominal and clausal complements of adjectives, as shown in (79) and (80), respectively.

(79) **Il est tout à fait* ***différent à*** *Maria.*
He is completely different to Maria.
Adjective complementation error between *différent* (different) and *à* (to).

(80) **Il est* ***libre à*** *choisir ce qu'il fait.*
He is free to choose that what he does.
Adjective complementation error between *libre* (free) and *à* (to).

On the other hand, we have an occurrence of overdetection in the corpus of simplified, error-full sentences. This is due to having other adjectives in the corpus, besides those which have legal complements. Moreover, we also tested the corpus after having corrected all the adjective complementation errors and we found six occurences of overdetection in this error category. These are mostly due, not to missing information in the lexicon, but to incomplete analyses from the parser or to incorrect analyses, leading to the erroneous attachment of a phrase as complement of the adjective. Thus, in (81), a CPA error is noted between the adjective *proche* and a tense phrase containing only *avancerait.*

(81) *Cet homme* ***proche*** *de la nature* ***avancerait*** *mais pas de la même façon qu'un homme social.*
This man close to the nature would-go-forward but not of the same way as a man social.
Adjective complementation error between *proche* (close) and *avancerait* (would-go-forward).

Adjective complementation errors are relatively hard to detect because most adjectives do not necessarily take complements and because of the well-known ambiguity of preposition phrase attachment. Indeed, a preposition phrase following directly an adjective does not necessarily act as a complement of that adjective, it might, for example, be a verb adjunct. The syntactic parser must necessarily try all attachment positions and the selection of the best analysis does not always correspond to the correct attachment of a given preposition phrase.

Verb complementation errors (CPV)

Verb complementation is an even more frequent error category than adjective complementation. Verb complements appear in most sentences and deciding whether a complement requires to be introduced by a preposition and selecting the correct one is a daunting task for language learners. Relying on one's own mother tongue is often a source of errors.

For this error category, for which controlling the information contained in the lexicon proved crucial, we have used a corpus of linguist sentences as well as a corpus of simplified sentences to conduct our tests. In addition, we verified for the verbs in both the linguist and the simplified corpora that the expected subcategorization frame was present in the lexicon.

Table 5.5: Linguist sentences with verb complementation errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	20	15	0	5	0
Percent	100	75	0	25	-

Table 5.5 shows a recall rate of 75% on CPV error category for linguist sentences, which is a good score. On three of the five cases in which the error was not detected, the erroneous complement, headed by a preposition, was taken to be an adjunct and attached to the sentence as an adverb phrase. In the remaining two cases, no complete analysis was provided by the system. When these linguist sentences were corrected and fed again into the system, only one sentence did trigger an occurrence of overflagging. Although the results are good, one must admit that part of the difficulty was avoided by using, in the linguist sentences, verbs which had only one subcategorization frame.

Results for simplified sentences, shown in Table 5.6 are less encouraging at first sight, as recall drops down to 27.5%. This is due in good part to verbs having several lexemes with different subcategorization frames.

Table 5.6: Simplified sentences with verb complementation errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	80	22	0	58	0
Percent	100	27.5	0	72.5	-

One should also note that the verb complementation error category can be divided into six subtypes with widely varying results. The six subtypes are missing prepositions, confusion of preposition, and adjunction of preposition, for both nominal and clausal complements. Example (82) gives an example of each of the subtypes. It is expected that both confusion of preposition and adjunction of preposition for nominal complements are not going to be detected by EDS, because the system considers such complements are adjuncts and happily attaches them as such. This prediction was confirmed by the tests, as attested in Table 5.7 presenting the details of the CPV error category by subtype. It is also interesting to note that results for clausal complements are always higher than for nominal complements.

(82)a. Missing preposition with nominal complement

**Cela bénéficiera la Grande-Bretagne.*
This will-benefit the Great-Britain

b. Missing preposition with a clausal complement
**Il choisit acheter un bonbon.*
He chooses buy a candy.

c. Confusion of preposition with a nominal complement
**Je pense d'eux.*
I think of them

d. Confusion of preposition with a clausal complement
**On cherche de continuer la recherche.*
One tries to pursue the research.

e. Supplementary preposition with a nominal complement
**Nous combattrons à l'ennemi.*
We will-figth at the ennemy.

f. Supplementary preposition with a clausal complement
**Chacun doit de ne pas dépasser la limite.*
Everyone must to NEG not pass-over the limit.

Reasons for the non-detection of the verb complementation errors are:

- the inherent capacity of the system to consider preposition phrases to be adjuncts rather than complements;
- the possibility to use a verb lexeme with a different subcategorization frame than the one expected given the sentence meaning;
- misanalyses which can sometimes put the verb and its complement into different chunks.

On the bright side, the system is reliable in that both on the error-full simplified sentence corpus and on a corrected version of it, there was no occurrence of overflagging of CPV errors. Thus, we can rather confidently say that, although far from all CPV errors are detected, when one is detected then it is a real error.

Euphony errors (EUF)

Euphony errors have the particularity that they are very local errors, strict adjacency between the words being necessary. It thus seemed inappropriate to use a simplified set of sentences as a test corpus, as the simplifications should not influence the detection of the euphony errors. Therefore, this

Table 5.7: CPV error breakdown on simplified sentences

	Errors	Diagnosed	Detected	Undetected	Overdetected
Missing preposition on nominal complements					
Number	18	10	0	8	0
Percent	100	55.5	0	44.5	-
Missing preposition on clausal complements					
Number	5	3	0	2	0
Percent	100	60	0	40	-
Confusion of preposition on nominal complements					
Number	29	2	0	27	0
Percent	100	6.9	0	93.1	-
Confusion of preposition on clausal complements					
Number	10	3	0	7	0
Percent	100	30	0	70	-
Adjunction of a preposition on nominal complements					
Number	10	0	0	10	0
Percent	100	0	0	100	-
Adjunction of a preposition on clausal complements					
Number	8	4	0	4	0
Percent	100	50	0	50	-

Table 5.8: Authentic sentences with euphony errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	25	19	0	6	0
Percent	100	76	0	24	-

error category was tested only on authentic sentences which have been spell checked. Results are summarized in Table 5.8.

Of the six errors which were not detected, two are due to unknown words and one to an incomplete analysis which separated the two elements into different chunks. One of the errors, given in (83), is the only occurrence in the corpus of a supplementary contraction. It is therefore highly possible that this particular subtype of error bypassed our vigilance during the implementation phase and is not treated by the system. Another reason could be that the word *du* (of-the) does not bear the appropriate lexical feature (non-elided form) in the lexicon. The cause of the undetection of the remaining two errors is so far unexplained.

(83) **Les deux côtés* ***du argument*** *peuvent être justifié.*
The two sides of-the argument can be justified

The euphony error category provides one of the best set of results, with a recall at 76% (moreover on authentic data which is supposedly more difficult than the other two types of corpora) and no occurrences of overflagging in the test corpus, that is a 100% precision rate. These excellent results might be partly derived from the locality feature of the error category.

Gender errors (GEN)

The gender of nouns is something native speakers do not have to pay attention to, so ingrained it is in them. For language learners, this is quite another matter. Indeed, the gender of noun is abstract in that there is no set of rules which can predict reliably the gender of a given noun. Some clues can be derived from the ending of the words, but they are by no means error-proof. Moreover, some homophones might have different genders depending on their meaning.[7] Thus, the gender of nouns must be learned and is a frequent source of agreement errors with determiners, adjectives, and past participle.

Tests were conducted on a corpus of 50 simplified sentences, each with a gender error, and the results are summarized in Table 5.9.

Table 5.9: Simplified sentences with gender errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	50	29	3	18	1
Percent	100	58	6	36	-

Of the 18 errors which were not detected, 6 imply a non-obligatory pronoun reference in the sentence, as in example (84) where the pronoun *elle*

[7] *poste* is either feminine when speaking about a post office, or masculine when a job position is meant.

(she) could make reference to someone not mentioned in the first part of the sentence, rather than to the masculine word *mouvement* (movement). 3 other errors are due to ambiguous words in the sentence which can take both genders depending on their meaning. For 3 sentences, incorrect analyses of the sentences were provided, thus hiding the gender error. For the remaining 6 errors, no clear reason for the undetection could be extracted.

(84) *Pendant que le mouvement a libéré la femme de la cuisine, elle risque de l'enchaîner au bureau.*
While that the movement has liberated the woman from the kitchen, she risks to her chain to-the office.

Given the data in this simplified corpus, there are at least two ways to compute a recall rate. The first one is to consider that there are truly 50 errors in the corpus and to take into account both diagnosed and detected errors. This gives us a recall rate of 64%. The second method is to consider that in those 50 errors, only 40 could actually be detected by the error diagnosis system. This number of 40 is reached by subtracting 6 errors which are referential problems, 3 errors due to ambiguous lexical items, and 1 error which is actually a euphony error[8] (and was included in the detected columns of Table 5.9). By doing this, we find ourselves with 31 diagnosed and/or detected errors of the 40 which were detectable. This gives us a recall rate of 77.5%.

Irrespective of the manner in which recall is computed, whether we reach a rate of 77.5% or 64%, the achieved score is honorable, in particular in the view of the very few occurrences of overflagging in this error category. On each of the versions of the corpus of 50 simplified sentences, with a gender error or corrected, only one instance of overflagging was presented. Thus, gender error seems to be also a very reliable error category.

Homonymy errors (HOM)

The homonymy error category is the only one which is diagnosed through the phonological reinterpretation technique. Results of tests on this error category are thus the only ones available for the whole technique and their

[8]While both words of example (85a) are feminine, one must use the so-called masculine possessive determiner for euphony reasons in front of words starting with a vowel (85b).

(85)a. **sa égalité*
its-feminine equality

b. *son égalité*
its-masculine equality

This error is correctly diagnosed by EDS as a euphony error, rather than a gender error, although it was tagged in the FRIDA corpus as a gender error.

importance is therefore increased. Tests were conducted using a corpus of simplified sentences containing 68 homonymy errors. The results are provided in Table 5.10. An example of a correctly detected homonymy error is given in example (86).

Table 5.10: Simplified sentences with homonymy errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	68	31	2	35	4
Percent	100	45.6	2.9	51.5	-

(86) **Que se soit en Russie ou en France.*
Whether itself is in Russia or in France.
Homonymy error on *se* (itself). Correction proposal: *ce* (it).

35 undetected errors might seem a very large number. They are due to two factors. The first one is that phonological reinterpretation can apply only if a first pass in the error diagnosis system provides only partial analyses. It so happens than in 15 sentences of this ungrammatical corpus, the error diagnosis system still manages to reach complete analyses before phonological reinterpretation is activated, as in (87a), with its complete analysis in (87b)). For those, phonological reinterpretation has not been activated and there was therefore no possibility to diagnose homonymy errors. Moreover, in 20 other cases, even with phonological reinterpretation activated, it was not possible for the system to provide a complete analysis of the sentences, as in (88). As it was decided to display the results of phonological reinterpretation only when a full analysis had been reached, in order to control the number of overflagging occurrences, it was not possible to show homonymy errors in these additional 20 sentences, although some might have been computed during the phonological reinterpretation phase.

(87)a. ****Sont*** *attractivité.*
Are attraction.

b. $[_{\mathrm{TP}}\ [_{\mathrm{DP}}\ \mathrm{e}\ ]\ [_{\bar{\mathrm{T}}}\ \textit{sont}\ \ [_{\mathrm{VP}}\ [_{\mathrm{NP}}\ \textit{attractivité}\ ]\ ]\ ]\ ]$

(88) **L'homme est le plus fort* ***mai*** *la femme est la plus belle.*
The man is the most strong May the woman is the most beautiful.
No complete analysis, even with phonological reinterpretation.

There were four occurrences of overdetection, which is higher than for most other error categories. However, in all 4 cases, the overdetection appears on a word adjacent to one which actually carries a diagnosed homonymy error. Thus, by stretching things a little, we could even consider that these overdetections are simply a form of detection of an error but of wrongly posing the diagnosis, thus indicating two locations instead of one. Also, 2 errors were correctly detected but misdiagnosed, in that an erroneous correction proposal was made. It is thus important to make the learner aware of the limits of the system and to warn them that the correction proposals are tentative only and need to be verified.

For this error category, there are also two ways to compute recall. The first one is to take into account all 68 errors contained into the test corpus. Given that 31 errors were diagnosed and 2 additional were detected, we reach a recall rate of 48.5%. Now, by considering only those sentences in which the diagnosis technique had a chance to discover a homonymy error, that is in disregarding the 15 sentences for which a complete analysis was found before applying phonological reinterpretation, we reach a recall rate of 62.3%. Although not as good as the recall rate for euphony errors, the present result is already fair.

Number errors (NBR)

The number error category, while strongly linked to the gender error category, does not depend on knowledge by heart on the part of the language learners. It is more a question of remembering to make the necessary agreements between the different elements of a sentence. While this is relatively easy on simple sentences, errors are more frequent when sentences become more complex, with intervening phrases, coordination and the like. This seems to be reflected also in the results of tests on linguist sentences (Table 5.11) and simplified sentences[9] (Table 5.12).

Table 5.11: Linguist sentences with number errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	32	21	9	2	0
Percent	100	65.6	28.1	6.3	-

By taking into account both detected and diagnosed errors, the recall rate for linguist sentences reaches the high score of 93.8%. Of the nine errors which were detected but not correctly diagnosed, five are classified as such because the error tag was attached to only one of the mismatched elements. The remaining four indicate a voice error instead of the expected number error. In those four sentences, the incriminated error is on the past participle of the verb *se laver* (to wash oneself) and one can thus wonder if the misdiagnosis is linked to reflexive verbs. For the two sentences for which

[9]One must remember that 'linguist' sentences are in fact simpler than 'simplified' sentences, the latter being derived from authentic learner productions.

the number error was not detected, the underlying analysis of the sentence was not correct.

Table 5.12: Simplified sentences with number errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	50	28	0	22	4
Percent	100	56	0	44	-

On the simplified sentences, recall is at 56%, which is significantly lower than for linguist sentences. Incorrect analyses leading to undetected errors were encountered in many sentences. Sometimes, the mismatched elements are found in separate chunks. At other times, an adjective, as in example (89a), is taken to be a noun, thus forming a kind of compound noun with the preceding noun (89b), instead of a complex noun phrase with an adjective (89c).

(89)a. **Il se trouvera prié de rester dans les limites fixées par le parlement européens.*
He himself will-fixed asked to stay within the limits fixed by the parliament european-plural.

b. * [$_{\texttt{DP}}$ le [$_{\texttt{NP}}$ parlement] [$_{\texttt{NP}}$ européens]]

c. * [$_{\texttt{DP}}$ le [$_{\texttt{NP}}$ parlement [$_{\texttt{AP}}$ [$_{\texttt{DP}}$ e_i] [$_{\texttt{A}}$ européens]]]$_i$]
Number error between *parlement* (parliament) and *européens* (european).

Often coordination leads to an ambiguity of attachment, which is an open door to misflagging. Not showing an error is sometimes preferred on the grounds of preventing overflagging occurrences. At times also, number errors on the verb of a relative clause remains undetected because the relative pronoun, which acts as the subject of the clause, is not correctly linked to its antecedent.

The not so good results for the number error category are partly linked to the desire to keep occurrences of overflagging in check as sometimes further analyses of the same sentence (recall that only one is actually presented to the users) do contain the appropriate error marks. As for the occurrences of overflagging already present in the corpus of simplified sentences, example (90) is due to an ambiguity in a structure where both mass and count nouns can be used and where the incriminated noun did not bear the appropriate mass noun feature and thus was required by the system to be a plural. The other occurrences are not explained so far.

(90) *... trop de pouvoir ...*
... too-much of power ...
Number error between *trop* (too-much) and *pouvoir* (power).

Negation errors (NEG)

The frequency of the negation error category is not known precisely as negation errors gather parts and pieces of several categories of the CECL typology. There are two main instances of negation errors: the omission of the negative particle *ne*, which is compulsory in written French but not orally, and the omission of *pas* (not). Both types are encountered in the corpus which is not surprising, given that in many languages only one element is necessary to indicate negation. The difficulty for language learners is to forget neither.

To test the error diagnosis system on negation errors, we used a corpus of 42 simplified sentences. The results of these tests are summarized in Table 5.13.

Table 5.13: Simplified sentences with negation errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	42	21	0	21	1
Percent	100	50	0	50	-

Recall is only at 50% which is not a very high score, especially given that one occurrence of overflagging happened in the original test corpus and 4 on the same corpus once the sentences were corrected. Main difficulties in detecting errors seem to be linked to incorrect analyses on the part of the parser, with the negative adverb *pas* (not) being attached to an incorrect position, often to the following verb as in (91), or taken to be a noun (meaning 'step') (92).

(91)a. **On peut pas répondre à la question.*
One can not answer to the question.

b. $[_{\texttt{TP}}\ [_{\texttt{DP}}$ *on* $]_i\ [_{\overline{\texttt{T}}}$ *peut* $[_{\texttt{VP}}\ [_{\texttt{TP}}\ [_{\texttt{DP}}\ \mathrm{e}_i\ ]\ [_{\overline{\texttt{T}}}\ [_{\texttt{VP}}\ [_{\texttt{AdvP}}$ *pas* $]\ [_{\overline{\texttt{V}}}$ *répondre à la question* $]\ ]\ ]\ ]\ ]\ ]\ ]$
One can not answer to the question

(92)a. **En étant pas si isolé.*
In being not so isolated.

b. $[_{\texttt{CP}}$ *en* $[_{\texttt{TP}}\ [_{\texttt{DP}}\ \mathrm{e}\ ]\ [_{\overline{\texttt{T}}}$ *étant* $[_{\texttt{VP}}\ [_{\texttt{NP}}$ *pas* $[_{\texttt{AP}}$ *si isolé* $]\ ]\ ]\ ]\ ]\ ]$
In being step so isolated

Most occurrences of the problem of the ambiguous *pas* (not/step) illustrated in (92) can be put aside by marking a strong preference for structures in which this particular word is a negative adverb, rather than a noun. As for the incorrect attachment of the negative adverb, it is, once again, due at least in part to the desire to reduce the number of overflagging occurrences, which tends to promote error-free analyses to the top of the computed analyses.

Adjective order errors (ORDAJ)

In French, adjectives are placed either before or after the noun they modify. The most frequent position is post-nominal, but some specific adjectives can take both positions, with a slight difference in meaning between pre-nominal and post-nominal positions. There are, however, truly post-nominal adjectives. Thus, we can detect post-nominal adjectives placed pre-nominally with a higher degree of certitude than we can detect pre-nominal adjectives placed post-nominally. Tests on adjective order were performed on a corpus of simplified sentences and the results are summarized in Table 5.14.

Table 5.14: Simplified sentences with adjective order errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	23	7	0	16	0
Percent	100	30.4	0	69.6	-

Some of the instances in which the error diagnosis system did not detect any adjective order error are due to adjectives which can take both pre-nominal and post-nominal positions, depending on the context, as in the two correct phrases of example (93). Errors with this type of adjectives are not flagged in order to prevent a too high number of overdetections.

(93)a. *Ces derniers mois.*
These last months.

b. *La semaine dernière.*
The week last.

In a few other cases, the ordering error occurs between two adjectives which should be inverted. Their usage is more akin to fixed phrases or compound words, as in example (94a), corrected in (94b). This type of adjective order error is not treated at present in EDS as fixed phrases were not part of the selected error categories. Further research would be needed to detect this type of adjective inversion.

(94)a. **Le produit brut intérieur.*
The product gross domestic.

b. *Le produit intérieur brut.*
The product domestic gross.

Given these good reasons for not detecting many errors, we reach only a rather low recall of 30.4% for adjective order errors. On the bright side, however, there is not a single instance of overdetection in either the corpus of simplified sentences nor in its corrected equivalent. Once again, users can be quite confident that when an error of this category is detected by the system, there is actually such an error in the input data.

Adverb order errors (ORDAV)

The diagnosis of adverb order errors suffers from the same kind of difficulties as adjective order errors. The placement of adverbs in French has a certain degree of freedom which must be accommodated in order to prevent overflagging. However, at the same time, there is a number of restrictions on which types of adverbs can be positioned in specific locations, taking into account that many frequent adverbs belong to several types at the same time.

Table 5.15: Linguist sentences with adverb order errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	19	7	0	12	0
Percent	100	36.8	0	63.2	-

The first series of tests on the diagnosis of adverb order errors were conducted on linguist sentences and the results are provided in Table 5.15. Recall is very low, at 36.8%, especially considering that this corpus contains carefully selected sentences and adverbs. It does not bode very well for tests on the second corpus, this time composed on authentic sentences, as one usually gets the best results with linguist sentences, then with simplified ones, and the worst with authentic ones.

Table 5.16: Authentic sentences with adverb order errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	36	6	0	30	2
Percent	100	16.7	0	83.3	-

Table 5.16 presents the test results for authentic sentences. Indeed, on authentic sentences, recall diminishes by more than 50% to reach the very low score of 16.7%. In order to defend the error diagnosis system, one must take into account that the sentences under scrutiny contain many other errors and can be very long (with a maximum of 55 words within a single

sentence). Moreover, it is to be noted that the two occurrences of ORDAV overflagging occur in a sentence which received only partial analyses, and whose error marks are therefore less reliable than if it had been completely analyzed. As such, precision stands at 75%.

Nevertheless, the poor results obtained on the adverb order error category can be partially explained by the numerous legal positions for adverbs in a sentence, the difficulty to classify adverbs into strict types which can then be assigned to specific positions, and the strong desire to minimize overflagging which leads EDS to refrain marking an error when in doubt and thus to lower recall rates.

Person errors (PER)

Person errors are not as frequent as errors of the two other types of agreement we have discussed, namely gender and number.[10] This error category was tested on a corpus of simplified sentences and the results are shown in Table 5.17.

Table 5.17: Simplified sentences with person errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	45	29	5	11	0
Percent	100	64.4	11.1	24.4	-

The five detected errors can be split into two groups. Three sentences receive a person error tag on only one of the mismatched elements, while the other two are considered to be homonymy errors.

Of the eleven undetected errors, six are referential errors and cannot be diagnosed without a complete semantic analysis, as in example (95a) where the error is not detected by the system as there is a plausible interpretation, while (95b) is the correction expected given the context of the sentence.

(95)a. *Ils disent: fais ce que je veux.*
They say: do that which I want.

b. *Ils disent: fais ce que tu veux.*
They say: do that which you want.

One other undetected error comes from a sentence containing other fatal errors which prevented the detection of the person error. The remaining four undetected errors where not tagged in the analysis presented to the user,

[10]Note that we consider an error between, say, a first person singular subject and a first person plural verb to be strictly a number error, and not a person error. An error between a second singular subject and a third plural verb will be taken as a double number and person error.

but where detected in the second computed analysis of those sentences, thus indicating that the diagnosis mechanism works well, even though the analysis selection mechanism is not without faults.

The rough recall rate is at 75.6%. However, if we subtract from the number of sentences the ones containing a reference error which we know not to be able to detect, and take into account both the correctly diagnosed errors and the detected ones, we obtain a recall rate of 87.2% which is a high score, especially compared to some for other categories which were much lower. Moreover, overflaggging occurrences are pretty rare, as there is only one instance in the corresponding corpus with corrected sentences.

Missing punctuation errors (OUB)

Errors of this category, which for EDS contains only errors on missing hyphens between a verb and its inverted subject, have not been tested. It was implemented on the basis of some linguist examples, but there is no occurrence of such an error in the FRIDA corpus extract containing errors on hyphens.

Redundancy errors (RED)

The efforts developed to treat a specific subtype of redundancy errors[11] seem at first sight to have be in vain. The specific error subtype we are trying to diagnose is the erroneous use of *des* (of-the) instead of *de* (of) in complex determiners as illustrated in the incorrect sentences (98) and (99).[12]

[11]This error category was not mentioned in chapter 4. Its inclusion derives from a late request from members of the FreeText project. Moreover, we attempt to diagnose only a very small subtype of redundancy errors.

[12]There seems to be a parallel between this error type and the problem of indefinitness encountered in the sentences of example (96). Both (96a), which is indefinite, and (96b), with a definite determiner and a relative clause, are grammatical. However, (96c), with a definite determiner but no relative clause, is ungrammatical.

(96)a. *Il est venu des amis.*
It is come some friends.

b. *Il est venu les amis que j'attendais.*
It is come the friends whom I awaited.

c. **Il est venu les amis.*
It is come the friends.

In (98) and (99), *des* (of-the) is the contraction of the definite *de les* (of the). If a relative clause is added, the sentence becomes grammatical, as in (97).

(97) *J'ai mangé assez des olives qui sont dans ce plat.*
I have eaten enough of-the olives which are in this dish.

(98) **Il s'est fait beaucoup des ennemis.*
He himself has made many of-the enemies

(99) **J'ai mangé assez des olives.*
I have eaten enough of-the olives.

Efforts seem to have been without results as none of the topmost analyses, for none of the 10 linguist sentences or the 22 simplified ones shows any error. Recall is therefore at 0%. Fortunately, there is a rather simple explanation. First of all, it is important to be aware that if the topmost analysis for a given sentence does not bear a RED error mark, a further analysis does for most sentences. Thus, the problem does not lie in the detection and diagnosis of the error per say, but rather on the ordering of the alternative analyses. As it has been mentioned a number of times already, we took the policy to favor a low number of overflagging occurrences over higher recall rate, implementing a device which promotes error-free analyses to the top. It is this same mechanism which prevents the redundancy errors to be shown. In most cases, analyses of sentences co-exist where the word which should be followed by *de* (of) instead of *des* (of-the) is taken either as a determiner or as an adverb. The error diagnosis mechanism is tailored only to the determiner analysis and thus misses when the word is analyzed as an adverb.

The first two analyses for example (99) are given here as (100a) and (100b). (100b) is the expected structure with the appropriate RED error mark, but it appears in second position and is thus not displayed to the users.

(100)a. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \textit{J'}\ ]\ [_{\overline{\mathtt{T}}}\ \textit{ai}\ [_{\mathtt{VP}}\ \textit{mangé}\ [_{\mathtt{AdvP}}\ [_{\mathtt{PP}}\ [_{\mathtt{AdvP}}\ \textit{assez}\ [_{\overline{\mathtt{P}}}\ \textit{des}\ [_{\mathtt{DP}}$
$[_{\mathtt{NP}}\ \textit{olives}\ ]\ ]\ ]\ ]\ ]\ ]\ ]\ ]\ ]$
I have eaten enough of-the olives

b. $[_{\mathtt{TP}}\ [_{\mathtt{DP}}\ \textit{J'}\ ]\ [_{\overline{\mathtt{T}}}\ \textit{ai}\ [_{\mathtt{VP}}\ \textit{mangé}\ [_{\mathtt{DP}}\ \textit{assez}\ [_{\mathtt{PP}}\ \textit{des}\ [_{\mathtt{DP}}\ [_{\mathtt{NP}}\ \textit{olives}\ ]\ ]\ ]$
$]\ ]\ ]\ ]$
I have eaten enough of-the olives
Redundancy error on *des* (of-the).

A possible solution to the current problem of not showing the expected analysis would be to promote the importance of the determiner reading of the words for which there is this type of redundancy error over the adverb reading. If this took precedence on the error-free priority currently implemented, the analyses containing the appropriate error marks would be displayed to the users, thus hopefully increasing the recall rate to a satisfactory level.

Voice errors (VOI)

The last error category to have been tested is voice. Two tests were conducted, the first on linguist sentences, and the second on simplified sentences. Results are presented in Table 5.18 and Table 5.19, respectively.

Table 5.18: Linguist sentences with voice errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	10	5	2	3	0
percent	100	50	20	30	-

On linguist sentences, the recall rate, taking into account both diagnosed and detected errors, reaches a score of 70% which is honorable. The two errors which have been detected but incorrectly diagnosed were considered to contain a homonymy error and a correction suggestion was made to change the verb for one of its homonyms. (101a) presents one of the incorrect sentences and (101b) the suggested correction.

(101)a. **Il se dort.*
He himself sleeps.

b. *Il se dore.*
He himself sunbathes.

Table 5.19: Simplified sentences with voice errors

	Errors	Diagnosed	Detected	Undetected	Overdetected
Number	8	2	1	5	0
percent	100	25	12.5	62.5	-

As often, results are worse on simplified sentences. Indeed, recall for detected and diagnosed sentences reaches only 37.5%. This is due in part to the fact that 3 out of the 8 simplified sentences do not receive a complete analysis from the system. Moreover, the separation of the different chunks appears at, or close to, the erroneous part, which in the three cases is the addition of an illegal reflexive pronoun, as illustrated in (102) where the supplementary reflexive is in bold.

(102) **Il y a une seule chance de* ***s'****adhérer au système.*
There here has one single chance to oneself adhere to the system.

For the other two sentences in which the mistake was not detected, it is this time a case of a missing reflexive pronoun with a pronominal verb. Why such missing reflexives are sometimes detected but not always has not found a satisfactory explanation so far.

Recapitulation

Table 5.20 gives a summarized overview of the different recall rates achieved for the fourteen error categories detailed above, depending on the type of corpus used.

Table 5.20: Recapitulation of recall rates in %

	Linguist	Simplified	Authentic
AUX	66.6	30.8	
CLA	50		
CPA		40	
CPV	75	27.5	
EUF			76
GEN		64 – 77.5	
HOM		48.5 – 62.3	
NBR	93.8	56	
NEG		50	
ORDAJ		30.4	
ORDAV	36.8		16.7
PER		75.6 – 87.2	
RED	0	0	
VOI	70	37.5	

As one can see from Table 5.20, results vary greatly from one error category to the next. On linguist sentences, we go from a rock bottom 0% recall to a high 93.8%. On simplified sentences, depending on the calculation method, one reaches 87.2%, but many recall rates are between 30% and 40%. Moreover, one must keep in mind that the number of errors on which these recall rates are computed vary greatly from a handful of sentences to a maximum of 80. There is no guarantee that these recall rates would remain constant on larger test corpora. It is thus difficult to give an overall quantitative appreciation of the system. We can nevertheless say that agreement errors (gender, number and person), as well as euphony, negation and homonymy errors, tend to perform better than the others.

5.2.2 Qualitative results

Assessing results qualitatively is much more subjective than doing so quantitatively. It is often more akin to a general appreciation than to a detailed review, although some specific points may be discussed for illustration purposes. In this section, we first discuss constraint relaxation and some of the error categories treated by this technique. We then continue with phonological reinterpretation, before looking at the system as a whole.

Constraint relaxation

Results from constraint relaxation vary greatly depending on the error categories one looks at, that is depending on the constraints which are relaxed. While results are very good for some error categories such as gender or euphony, they are poor for others like verb complementation or adverb order on more authentic data. It is quite clearly in the more complex error categories that one encounters more difficulties, which is not very surprising.

An important factor in the quality of the results of a given error category is naturally the information contained in the dictionary. To obtain good results, the information must obviously be correct and complete. But apart from this, the information must also be relatively simple and with as little alternatives and ambiguities as possible.

For verb complementation (CPV) errors, for example, the difficulty of error checking greatly increases when there are several subcategorization frames for a single verb.[13] The system must check each subcategorization frame to find a best match in which all complements are present and none superfluous, or against which to mark the diagnosis. Moreover, lacking any form of semantics in the system, we are not able to determine whether some prepositional phrases should be considered as complement or as adjunct. Therefore, all prepositional phrases which do not find an appropriate slot in the subcategorization frame end up as adjunct and are not treated as an error. Sentences in (104a) and (104b) are treated in exactly the same way and the system misses the CPV error in (104a).[14]

(104)a. **J'aide **à** la personne.*
I help to the person.

b. *J'aide à la maison.*
I help in the house.

Adverb order errors are also among the less well treated error categories. There is, first of all, a certain degree of freedom in the position of adverbs in French sentences. Then, it is quite difficult to strictly classify the adverbs along the lines of their legal positions in a sentence. Moreover, it can be argued that some adverbs belong to several types depending on their alternate meanings. It thus becomes quite difficult to ascertain automatically whether a given adverb is in one of its legal positions or not and the results

[13]Having several subcategorization frames is the norm rather than the exception.

[14]However, if the complement is pronominalized, as in (103), the error is correctly diagnosed.

(103) **Je lui aide.*
I him/her help

of such an error category are less than satisfactory. Both overflagging and non-detection occur much too frequently in this error category.

There are, nonetheless, other error categories for which results are of a much higher quality. It is the case of euphony errors, for example. Errors in this category are 'simpler' to detect. The words involved must be adjacent to one another. Some very specific word features, used only for euphony matters, are tested. There is little room for ambiguity and none for semantics, which suits an automatic syntactic system well. Overflagging in this error category is rare and undetected errors are often due to words not belonging to the dictionary, such as foreign or borrowed words. This confirms the importance of the correctness and completeness of the dictionary one uses.

Although some error categories are more difficult to diagnose with constraint relaxation than others, the overall result of the use of constraint relaxation as a technique for grammar error diagnosis is indeed positive if proper care is respected in the implementation.

Phonological reinterpretation

Qualitatively speaking, each instance of phonological reinterpretation can be either very good or very bad. One either indicates a true homonymy error or one overflags. This might not seem so different from other error categories. However, as one proposes a correction with phonological reinterpretation, overflagging appears quite badly because in those cases the correction proposal is usually completely off the mark, clearly showing a real misinterpretation of the sentence. Now, when phonological reinterpretation indicates a true homonymy error, the correction proposal can also be appropriate or not. However, an inappropriate correction proposal for a real error is not as blatant, especially if the proposal is of the correct lexical category and the choice between the alternatives is a question of semantics.

The number of reinterpreted words in a sentence is clearly linked to the number of chunks resulting from a parse of that sentence. Thus, a sentence analyzed as many chunks has a higher risk of having several of its words reinterpreted. However, it does not necessarily correspond to a similar increase in the risk of the users employing homophones. Therefore, in a sentence analyzed as many chunks, the risk of overflagging is greater than in a sentence analyzed as fewer chunks. In order to reduce the number of overflagging occurrences, the decision was taken to present the diagnosis by phonological reinterpretation only if it led to a complete analysis. Indeed, if phonological reinterpretation still resulted in chunks, it would be much easier for correct words at chunk boundaries to be reinterpreted erroneously.

Through phonological reinterpretation, several error categories can be detected, but so far they are all diagnosed as homonymy errors.

(105) **Heureusement elle a **réussit** à le retrouver.*
Luckily she has succeeds to it find-again

In example (105) above, *réussit* (succeeds) is an indicative present, third person singular, instead of a the past participle *réussi* (succeeded). It is a tense/mode error which we do not treat as such in the error diagnosis system, but the mistake is detected through phonological reinterpretation and, so far, given a homonymy error mark. It would be a great improvement to refine the homonymy error diagnosis by looking in more details at the common features of the original word and the reinterpreted one. In the present case, it would be easy to realize that we are dealing with the same verb, only at a different tense and mode, and to mark the error as such.

The system as a whole

Not all errors in a sentence or a text are diagnosed by the system. Some errors might not be part of the set of treated error categories. Moreover, not all errors which should be treated are found by the system. It is therefore absolutely necessary to warn the users of this matter, so that they do not believe that the system in infallible and detects all errors.

Indeed, some errors in the sentence, whether they are themselves part of the set of treated errors or not, prevent the diagnosis of other errors in the same sentence. By correcting the errors flagged by the system and by rechecking the syntax, one might see other errors appear which were, in fact, hidden before by the previous errors. Sometimes, the blocking error is not part of the set of errors treated by the system. This is the case in particular with missing diacritics on the final 'e' of past participles from verbs of the first group (ending in '-er').

(106)a. **Mais soudainement j'**ai** tomb**e**.*
But suddenly I have tomb.

b. **Mais soudainement j'**ai** tomb**é**.*
But suddenly I have fallen.
Auxiliary error on word *ai* (have).

c. *Mais soudainement je **suis** tombé.*
But suddenly I am fallen.

In example (106a), the missing diacritic on *tombe* (tomb) is not detected by the spell checker because it results in a legal word. The auxiliary error is not detected either. However, if the missing diacritic is replaced, as in (106b), the auxiliary error is appropriately detected by the system. The correct sentence is given in (106c).

If it is of the utmost importance that the users of the system be aware that not all error can be diagnosed by the system, and that, even for the error categories supposedly treated, not all error are found each time, it is equally important for them to know that not all flags raised by the system denote real errors. Indeed, there is a certain amount of overflagging occurrences, which vary depending on the error category. Qualitatively speaking, one must find the right balance between recall and precision. A high recall rate indicates that a large proportion of the errors have been found, while a high precision rate confirms that the system detects real errors and does not overflag much. While ideally both should be high, this is rarely the case. With language learners, one must tend towards a high precision rate, be it at the cost of a lower recall rate. Indeed, it is more acceptable for the system to leave some errors undetected than to overflag so much that one does not know anymore which error diagnosis denotes actual errors in the learners' productions.

However, overflagging is a major unresolved issue within EDS when it is used on authentic sentences. The longer the sentence, the higher the risk of having overflagging instances. Some error categories seem to be more prone to overflagging than others, but all are concerned by the phenomenon. Some overflagging occurrences are due either to incorrect information in the lexicon or to incomplete or imperfect implementation of the diagnosis techniques.

(107) *... beaucoup d'attention ...*
... much of attention ...
Number error between *beaucoup* (much) and *attention* (attention).

For example, in (107), there is an overflag indicating a non-existant number error between *beaucoup* (much) and *attention* (attention). It is either a coding error with a too restrictive constraint which does not take into account that *beaucoup* (much) can take a singular complement if this complement is a mass noun, or to an incomplete entry in the lexicon for *attention* (attention) where the mass noun feature is missing. In both cases, the problem can be easily put to rights.

More problematic are the occurrences of overflagging resulting from an incomplete or erroneous analysis of a sentence. In these cases, it is much more difficult, if not to say impossible in some instances, to pinpoint the precise reason of the overflagging and to correct it. Numerous real errors in a sentence can prevent the system to reach a good analysis of the sentence and even partial analyses can be themselves faulty. Misinterpretation of ambiguous attachments, coordinations, or words is a high risk for overflagging. This risk can be slightly mollified if one takes the option to warn the users each time a sentence does not attain a complete analysis, so that they know that in these cases they must take the diagnosis as tentative only and not trust it as completely as at other times. The error flags would still be raised, but they could be displayed as warnings to the users. Another possibility to attempt to contain the number of overflagging occurrences is to be stricter in the way constraints are relaxed. This would probably diminish the number of real errors detected by EDS, but might be beneficial in the long run if it significantly lowers the number of overflagging occurrences. This should be investigated at least for error categories with few errors in the corpus and a high percentage of overflagging.

Results obtained with the error diagnosis system vary depending on the testing corpus. On linguist sentences, one obtains a good recall for most error categories, and very few instances of overflagging, even when all errors in the sentences have been corrected. Results are not as good with simplified sentences, but they are still more than acceptable. This shows that the error diagnosis system has the capacity to detect and diagnose the different error categories, even in linguistically complex sentences. As expected, results are worse with authentic data, due in good part to the length of sentences and to the numerous errors they contain. Some errors, which our system is not expected to treat, can prevent the detection of some other errors, of categories which are usually correctly treated in simpler settings. This decrease in quality of the diagnosis on more complex corpora was expected and does not surprise us, nor prevent us from finding the system to be useful on such corpora.

Looking at the results on the authentic FreeText corpus (see appendix B.3.2), one can note that, for some error categories implemented in EDS, no error was found. This is due in part because the corpus is quite small and not all error categories are represented in it, but also partly because a number of errors are not detected by the system. One seems to detect a fair share of euphony and number errors, without too many instances of overflagging. On the other hand, for ORDAV and CPV error categories, there seems to be more false detections than correct ones. Flags from these error categories might be very disturbing to the language learners if their precision is too low. Restricting, or even removing, the error diagnosis mechanism for some poorly treated error categories might benefit the system as a whole. It would enable the system to achieve an overall higher precision rate with the disadvantage to lower, or suppress completely, the diagnosis of some error categories. The resulting system would hopefully be less confusing to the users than a system in which occurrences of overflagging are the norm.

5.2.3 Efficiency

Being based on the Fips syntactic parser (see section 4.1), the efficiency of the error diagnosis system is linked to the efficiency of the parser. However, it is not as efficient as the original Fips, given the transformations to which it was submitted in order to transform it into an error diagnosis system.

Both constraint relaxation and phonological reinterpretation have an adverse effect on efficiency.

Constraint relaxation

Constraint relaxation diminishes the efficiency of the parser because of the increased number of alternatives which need to be treated. Every time a constraint is relaxed, a constituent which would not have existed otherwise must now be taken into consideration. Given that this applies as well for all of its subconstituents, one imagines very well the explosive nature of this combinatorial process.

(108)a. *Tu suis le guide.*
You follow/am the guide.

b. $[_{\texttt{TP}}$ $[_{\texttt{DP}}$ *Tu*] $[_{\overline{\texttt{T}}}$ *suis* $[_{\texttt{VP}}$ $[_{\texttt{DP}}$ *le guide*]]]]
You follow the guide

c. * $[_{\texttt{TP}}$ $[_{\texttt{DP}}$ *Tu*] $[_{\overline{\texttt{T}}}$ *suis* $[_{\texttt{VP}}$ $[_{\texttt{DP}}$ *le guide*]]]]
You am the guide
Person error between *Tu* (You) and *suis* (am).

If constraint relaxation was not activated for person agreement, there would be one reading of the simple sentence in (108a). This reading is given in (108b). With this constraint relaxed, however, a second reading is possible when the system follows the alternative with a person error between subject and verb, as in (108c). For the sake of the example, we chose, in (108a), a simple sentence with a grammatical reading. There is an even greater amplification of number of 'new' constituents for ungrammatical sentences which, instead of being rejected by the system at an early stage, end up with one or several complete analyses. Each new alternative has its toll on the efficiency of the whole system. Therefore, efficiency diminishes with each new error category that one wants to treat in the system, as each category corresponds to the relaxation of one or several constraints.

The constraint relaxations needed for the fourteen error categories treated by this technique also meant an increased number of complete analyses resulting from the parsing process. We had to devise an analyses selection mechanism, as described in section 4.3.4, in order to retrieve the most appropriate analysis given the CALL context of the error diagnosis system. This selection mechanism also takes its toll on the efficiency of the diagnosis system.

With fourteen error categories diagnosed by constraint relaxation, and the additional analyses selection mechanism, the error diagnosis system is indeed slower than the original Fips parser. But is is still good and fast enough for the users not to notice the difference on most sentences.

Phonological reinterpretation

Phonological reinterpretation is another story and is not an efficient diagnosis technique at all. First of all, it requires a first pass in the system in order to obtain chunks and identify the candidate words for reinterpretation. The system is therefore at least as slow as the first pass in the system. Then comes the time for dictionary look up. For each candidate word, this is done in two stages: (i) looking up the phonetic representation of the word, and (ii) retrieving the homophones. Dictionary look up is not the fastest action, but indexed access prevents it from being too long. Once the words are inserted in the chart, a second phase of analysis is started, by nature longer than the first one as new alternatives were added in the form of the reinterpreted words. The diminished performances naturally depend on the number of chunks resulting from the first pass, that is of the number of candidate words for reinterpretation. They also depend on the number of alternative homophones found for each reinterpreted word.

There is no doubt theoretically that phonological reinterpretation is a particularly inefficient technique. In practice, the problem was even worse than expected. Indeed, on some very long sentences, all the resources of a Pentium II 400 MHz PC were used up by the process which had to be terminated before it could give a result. In view of this, we decided to turn off constraint relaxation when phonological reinterpretation was activated, so that there were fewer alternatives to be treated at the same time. This expedient improved efficiency somewhat. However, it means that only homonymy errors can now be diagnosed during the phonological reinterpretation pass, preventing the errors diagnosed by constraint relaxation to appear. Moreover, to avoid too many instances of overflagging of the homonymy error, only a pass resulting in a complete analysis is taken into account for display to the users. This limits results obtained by phonological reinterpretation even more.

There are certainly ways to improve the efficiency of the phonological reinterpretation technique with a better implementation. In the present implementation, alternative words are simply added in the chart of constituents which already contains the results of the first parse and the analysis process is started again. A better control of the content of this chart and some sort of cleaning up between the two passes is probably called for.

5.3 Comparison with another grammar checker

In order to evaluate the interest of our error diagnosis system, we decided to compare it to another grammar checker. It would have been most interesting to choose a system described in chapter 2, but this proved impossible because the research prototypes described are not readily available. Thus, we turned to what seems the most common grammar checker, that is the grammar checker for French included into Microsoft®Word 2000 (thereafter shortened to WORD). As information on the inner workings of this commercial checker is not public, we had to use the system as a black box and could only make guesses at its internal processes.

For the comparison, we used the corpus of authentic sentences entered by student testers of the FreeText software (see appendix B.3.2). Naturally, spelling errors have been corrected, in order to be fair to EDS which does not incorporate a spell checking component while WORD does.

In this section, we first present test results on error categories which are treated by both systems. This allows us to concentrate first on really comparable data. We then discuss differences in the types of errors detected by both systems to contrast them and try to see if one is more appropriate than the other for use with language learners.

5.3.1 Comparison

We start our comparison by looking only at error categories which are diagnosed by both our error diagnosis system and the grammar checker included in WORD. While EDS provides the user with an error category for each detected error, WORD presents a more user-friendly feedback in the shape of sentences. The information contained in the sentence feedback was used to deduce the category of the detected error. We evoke the problem of overflagging before concluding with some comments.

Table 5.21 presents the number of correct flags obtained by our error diagnosis system and by WORD on the test corpus.

Table 5.21: Comparison of correct flags

Errors	Corpus	EDS	WORD
GEN	28	21	22
HOM	12	4	2
OUB	10	1	4
PER	7	4	6
NBR	7	4	4
EUF	5	4	4
AUX	4	1	3
NEG	1	0	1
Total	75	39	46

We must first of all note that neither system detects all the errors in the corpus belonging to these eight error categories. Then, it appears that, on rough numbers, WORD does slightly better than EDS. We must nevertheless be aware that errors were detected at one go with our error diagnosis system, while WORD, showing multiple errors in a sentence one by one, benefited from corrections enabling the system to detect and display subsequent errors in the sentence.[15]

One recurrent error, given in (109) was diagnosed differently by both systems.

(109) **grand mère*
big mother

WORD diagnosed this error as a missing hyphen '-' in between the two words. We therefore considered it to be part of the OUB error category. Our diagnosis system, on the other hand, diagnosed a gender error between *grand* and *mère*. These errors were thus tagged as GEN, rather than OUB. Although there is a certain degree of ambiguity in between the two interpretations, in context a human corrector is more likely to consider this error as a lexical problem, the missing '-', than as an agreement error. Ideally, this type of error should be treated at the lexical level by a spell checker rather than by a syntactic checker. However, as both words belong to the dictionary, it is doubtful how this could be achieved by any standard spell checker, as they usually just check that each word they encounter belongs to their dictionary. EDS detects only one missing hyphen in the test corpus, but it has been trained on missing hyphens between inverted verb and subject only, while in the data they are missing mostly on structures like example (109) and (110).

(110) *ce jour-là*
this day-there

A large number of gender errors is detected by both systems, although each checker misses about one fourth of these errors, which are the most frequent of the corpus.

WORD detected two homonymy errors. In both cases, it is a missing accent on *a* (has/to). For this error category, our diagnosis system performed better and detected four errors. One *a*/*à* (has/to) error is detected on a different sentence than WORD. Other detected errors are the use of an infinitive for a past participle of a verb of the first group (example (111)); use of a verb instead of its corresponding noun (example (112)); and use of a noun instead of an adverb (example (113)).

(111)a. **la crème glacée empoisonner*
the cream iced poison-verb-infinitive

[15]It is impossible to know if several errors are found at the same time but displayed only one at a time or if WORD stops checking after the first detected error.

b. *la crème glacée empoisonnée*
the cream iced poisoned

(112)a. **le directeur de la filme*
the director of the-feminine film-verb

b. *le directeur du film*
the director of-the movie

(113)a. **trot vite*
trot fast

b. *trop vite*
too fast

These examples clearly illustrate the potentialities of the HOM detection within our diagnosis system and the limitations of commercial checkers in this area. WORD seems to be able to detect homonymy errors only for specific known problematic words which are easily confused by native and non-native speakers alike, such as *a/à* (has/to). On the other hand, our error diagnosis system is capable of detecting the use of rarer homonyms or identical sounding versions of the same word, which is a type of error more likely to be made by language learners than by native speakers. Thus, on homonymy errors, our diagnosis system performs better than WORD both quantitatively and qualitatively.

As both systems detect an similar quantity of number (NBR) and euphony (EUF) errors in the corpus, one could legitimately wonder whether they are the exact same errors. In fact, they do not overlap precisely. In both cases, three of the four detected errors are the same, while the fourth error is different for both checkers. This kind of partial overlap of diagnosed errors in the different error categories is even more flagrant when the number of detected errors is not identical for both systems. For auxiliary errors (AUX), the single error detected by our diagnosis system is not found in the three discovered by WORD. The inner workings of the checkers cannot therefore be identical.

Of the seven agreement in person errors found in the corpus, four are detected by EDS and six by WORD. Moreover, the one error which is not detected by WORD, in example (114), is diagnosed by EDS. In (114), the detection of another error by WORD prevents the diagnosis of the person error.

(114) **Depuis ce jour je n'ai plus avoir une robe favorite parce que je n'aime pas le chagrin quand je deviendra trop grand!*
Since that day I NEG have anymore have a dress preferred because I NEG like not the sorrow when I will-grow-3rd-person too tall.

Unlike in other cases, the detected error was not corrected because the diagnosis was faulty, indicating that a missing past particple after *avoir* (have). Given the erroneous diagnosis, it did not seem legitimate to amend the sentence, with the resulting effect that no attempt was made by WORD to display feedback for other errors in the sentence. As for EDS, undetected errors are due to the different consituents belonging to distinct chunks in two cases. The third case, more problematic because more general, is due to a difficulty in assigning the proper referent to relative pronouns in incorrect sentences, leading to misdiagnosis between the relative pronouns and the constituents they must agree with.

WORD correctly diagnoses the negation error in (115), but EDS does not. Although EDS supposedly detects negation errors, the range is limited to the omission of either the negative particle *ne* or the negative adverb *pas* (not), when they should be used in pair. The only negation error in the corpus is in a structure with a negative determiner *aucun* (none), which is not taken into consideration by the current implementation of the constraint relaxation.

(115) **... j'entendais aucun de ses mots ...*
...I heard none of his/her words...

These differences in coverage of the errors are a very strong indication that the two systems do not use the same strategies to detect errors. If they did, the overlap should be complete, or at least larger. We now turn to elements which are specific to each checker.

5.3.2 Specifics of each grammar checker

Besides the error categories mentioned in Table 5.21, each system diagnoses a specific range of error categories. WORD signals one tense error, errors on the past participle (either supposedly missing, or the use of a wrong mode), one error on *tout* (all) emphasizing its dual status as determiner and adverb,[16] errors of capitalization and punctuation (although there are many instances of overflagging in the latter category). This error range keeps within the domain of likely errors for native language learners, as WORD never purports to cater to the needs of language learners.

[16]It is difficult to know where to classify the error in example (116) between the class, homonymy and number error categories. There is an agreement error which depends on the lexical category of the word which is often confused because both versions are homophonous.

(116) *tous puissants*
all-plural powerful

Our error diagnosis system, on the other hand, does not detect these error categories. Tense is considered too semantic for a syntactic checker, likewise missing words are a real handicap to our system and implementing the possibility of missing words of any kind in the sentence is like opening Pandora's box in terms of overgeneration of structures, as it is possible to imagine missing words almost anywhere. Capitalization was not a matter which preoccupied us and punctuation rules are often too subjective in French for errors to be detected with high enough recall and precision rates, as attested by the high number of overflagging occurrences in WORD, thus they were not dealt with.

While our system does not detect some error categories treated by WORD, there are also categories which are taken into account by EDS but not by WORD. These are errors of verb and adjective complementation as well as errors of adjective and adverb order. These error categories are typically not encountered in the writing of native speakers while they are a difficult matter for language learners.

(117)a. **... j'ai commencé réfléchir sur la liberté ...*
... I have started think on the freedom ...

b. ... I started thinking ...

In example (117a), the verb *commencé* (started) takes a sentential complement which should start with the preposition *à* (to). This preposition is absent from the learner sentence, perhaps by a negative transfer from the English sentence in (117b). This type of error is typical for language learners and is correctly diagnosed by our diagnosis system, showing its specificity as a tool for language learners.

5.3.3 Overflagging

Both grammar checkers overflag, that is, indicate errors on perfectly correct words. While overflagging is a known major problem for EDS given the high number of overflagging occurrences it produces, WORD is not exempt of such occurrences, to a lesser extent however.

(118) *Un jour, mes parents m'ont prise au Wallibi.*
One day, my parents me have taken-feminine to-the Wallibi

The sentence in (118) is grammatical as long as one considers that the narrator of the sentence, represented by the elided pronoun *m'* (me) is feminine. EDS does not detect any error in this sentence. WORD, however, indicates a gender error on the word *prise* (taken-feminine), forcing a masculine interpretation of the narrator of the sentence, for which there is no justification locally or in the wider context of the corpus.

Both systems encounter problems with missing apostrophes which lead them both to overflagging (or at least misflagging).

(119) **Quand j étais petite j aimais nager.*
When I-missing-apostrophe was small-feminine I-missing-apostrophe liked swim.

In (119), both systems seem to have trouble interpreting the single *j* as the elided form of the pronoun *je* (I), without its apostrophe. Therefore, WORD takes the word *étais* as a noun (stays) and diagnoses a number error between it and *petite* (small-feminine).[17] Our error diagnosis system, on the other hand, takes the single *j* as the letter 'j' and detects a person error between it and *aimais* (liked), as the letter 'j' is considered 3rd person singular, while *aimais* is 1st or 2nd person singular.[18] Missing apostrophes should be relatively easy to treat, perhaps through some preprocessing or *ad hoc* measure, and should slightly reduce the number of overflagging occurrences.

5.3.4 Remarks

Neither EDS nor WORD detect all the errors in the corpus. The users must be warned that a checked and amended text does not mean that it is completely error-proof.

The manner in which the corpus was tested with both checkers may have been unfair to EDS by according an advantage to WORD in correcting errors as they were flagged. This was the only manner to obtain information on sentences containing several errors, but it allowed WORD to re-check a less error-full, and therefore less difficult, sentence, while EDS was given only one chance. Whether this was a real advantage or not depends on whether it is only the display of multiple errors which is delayed in WORD (in which case there is no specific advantage) or whether it is the whole checking process which is restarted.

Both systems overflag, although not to the same extent. Overflagging is a real problem, especially for EDS, both because there are more occurrences, but also because its target public is less well equipped to deal with overflagging, not necessarily being able to recognize overflagging occurrences as such.

On error categories common to both EDS and WORD, WORD achieves similar or better results than EDS, except on one. Given the person-months which must have been devoted to both system, this overall result is not surprising. The exception is the homonymy error category, in which errors are

[17]A gender error should also be detected in that frame of mind, but is not. A possible explanation is that the number error must first be corrected before WORD can diagnose and/or display the gender error.

[18]Interestingly, no error is detected in the first part of the sentence where one would expect the same kind of overflag.

more specific to language learners. Errors of complementation and of word order also reinforce the fact that EDS is more specifically geared towards the needs of language learners.

5.4 Conclusion

The major challenges encountered by our error diagnosis system and discussed in this chapter are the overgeneration of structures and a latent lack of efficiency. These two hurdles should be resolved, at least in part, in order for EDS to become a viable, large scale, unrestricted grammar checker.

Overgeneration of structures has already been partially tackled with the adjunction of the selection mechanism. This mechanism, however, works by eliminating improbable structures after the analysis phase is achieved and all possibilities have been computed. It is therefore not helpful on the efficiency issue for which it would be more adequate to prune highly implausible partial analyses during the parsing process.

To tackle both overgeneration and efficiency at the same time, one should perhaps investigate in more details the possibility of relaxing constraints by packets. This possibility has already been evoked in sections 3.1.9 and 3.1.10, but has neither been pursued nor tested so far. Restraining the set of constraints relaxed at the same time would undoubtedly diminish the number of alternative analyses produced. However, the trade-off will be the number of passes needed to detect all error categories. Whether this would bring a heavy toll on efficiency because of the multiplication of passes, or whether it would be on the whole lighter because each pass would be much more efficient than the present single relaxed one can only be ascertained through testing.

If the multi-pass option proves interesting, it might be worth imagining lighter techniques than either constraint relaxation or phonological reinterpretation to diagnose some specific error categories. Indeed, a full syntactic analysis is not necessary to treat some error categories. The euphony error category, with its mandatory adjacency feature and its lack of influence on both sentence structure and meaning, is a good candidate. A surface scan of the sentence could locate words which are, or can be, elided or contracted and check whether the following word bears the corresponding elision or contraction features.

Examination of the FreeText data also informed us as to the numerous errors which were not supposed to be, and were not, treated by EDS. These errors, while a problem in themselves for language learners, are an issue for the system as well because they often prevent a good sentence analysis and therefore a good error diagnosis. We are thinking especially of missing diacritics resulting in legal words (in particular on past participles, as presented in example (106), page 183) and of near homophones such as exemplified in

(120) and evoked in section 3.2.7.

(120) a. *qui/qu'il*
who/which he

b. *et/est*
and/is

Nevertheless, there are very positive aspects to the results described in this chapter. Both constraint relaxation and phonological reinterpretation are theoretically viable diagnosis techniques. This view is supported by the good results obtained for the vast majority of the error categories tested on linguist sentences, as well as by the results obtained for homonymy on simplified sentences. This supports us in the view that the well-known constraint relaxation technique and the more innovative phonological reinterpretation technique are of great interest for grammar error diagnosis. Once the diagnosis of errors detected by phonological reinterpretation is refined, the importance of combining techniques in order to cover a larger range of error categories will be made even more obvious than it is now. Although there is a small range of overlapping error categories, constraint relaxation and phonological reinterpretation mostly target different error categories. Eliminating either technique would be detrimental to the error diagnosis system as a whole by diminishing its coverage.

While some improvements are needed to turn the error diagnosis system into a full-fledged, large scale, independent, commercial grammar checker, the current prototype demonstrates its potential for language learners. The choice of the targeted error categories, the use of phonological information for diagnosis, as well as a non-prescriptive feedback respecting users' intentions are the main features which make this error diagnosis system especially well-suited for use by language learners.

Chapter 6

Conclusion

In this dissertation, after stating the problematic of grammar checking for CALL in the introduction, we have recalled the state of the art of grammar checkers and of the different diagnosis techniques employed. Then, we proposed a theoretical description of three diagnosis techniques for such a purpose and ways to use them in combination. The implementation of the theoretical ideas was then described and results of tests conducted with the error diagnosis system were provided.

The present chapter concludes this dissertation and is structured as follows. Section 6.1 exposes the contributions to the research domains of CALL and NLP. Section 6.2 recalls the interest of combining together several error diagnosis techniques. Section 6.3 states the limitations of the work accomplished so far. Section 6.4 proposes leads for further research and a final remark is made in section 6.5.

6.1 Contributions to the research domains

Work performed for this dissertation is spread over two main research domains: computer assisted language learning and natural language processing. Contributions have been made to both domains.

In the domain of CALL, we have showed that grammar error diagnosis can be performed successfully on free learner productions. This opens the way for more creative and communicative writing tasks for which language learners can receive automatic intelligent feedback on the grammaticality of their productions. Because our error diagnosis prototype is not yet very efficient on long sentences, using it in a CALL software, therefore in a restricted context, can only help it to achieve better performances. Even without tailoring the diagnosis system to specific CALL exercises, the writing medium and environment tend to make the productions shorter and thus easier to treat by an automatic error diagnosis system.

In the domain of NLP, the work carried out has proven the feasibility of adapting and transforming an existing syntactic parser in order to obtain a grammar checker. The well-known constraint relaxation technique has been pushed further than the toy prototype stage: it has been implemented on a large scale system with many constraints relaxed at the same time, enabling it to detect and diagnose errors of fourteen distinct error categories. An innovative diagnosis technique, phonological reinterpretation, has been successfully implemented. The implementation of both diagnosis techniques also served as a reminder that, in order to create a usable system, one must find ways in which to keep the overgeneration of structures and analyses at bay.

In section 6.1.1, we comment on the usability and competitiveness of our error diagnosis system. Section 6.1.2 recalls how the system incorporates innovative techniques for grammar error diagnosis. Section 6.1.3 briefly recapitulates the advantages and disadvantages of adapting a existing syntactic parser into an error diagnosis system.

6.1.1 Usable and competitive system

Our error diagnosis system is usable in terms of its large coverage of the French grammar and of its robustness, both derived from its underlying parser. It is also usable by language learners because it is specifically tailored to their needs. EDS supposedly does not accept ungrammatical French sentences without warning the users about their ungrammaticality. Specific categories of errors tailored to language learners are diagnosed by the system and feedback as to the type and location of the detected errors are provided. The treated error categories encompass categories common to both native and non-native speakers alike, such as agreement errors in number, gender, and person, but error categories specific to language learners, such as adjective order or verb complementation, are also present. Thus, EDS truly caters to its users' needs in terms of error coverage.

Although the system is usable in its present state, works remain to be done in order to lower the number of overflagging occurrences. We believe that the system's usability is greatly improved when its users are duly warned about its imperfections, that is, understand that some of their errors might not be detected by EDS and that EDS might also indicates errors where there is none. "It is important to inform the students that they should not implicitly trust grammar checkers in other matters but, rather, should always try to evaluate their advice. Students also need to understand that they should not assume grammar checkers detect all errors" (Jacobs and Rodgers 1999, p. 523). Users are intelligent and the goal of our error diagnosis system, in its present version, is not to correct their texts, but to make them think about what they have written, providing them with the clues needed to better their productions.

The error diagnosis system is also competitive with regards to commercial grammar checkers. The range of treated errors, tailored to language learners, makes it better suited than grammar checkers designed for native speakers and available as off-the-shelf products. The comparison between EDS and WORD (section 5.3) has shown the interest of EDS for language learners over a non-specific tool. As to the different systems which have been presented in chapter 2, sections 2.3 to 2.7, most did not go beyond the toy prototype stage. Moreover, they were often limited in the range of productions they could treat, such as CLEF (section 2.3.2); were not designed for language learners, such as the EPISTLE/CRITIQUE system (section 2.6.2); were limited in their coverage of the grammar and in the size of their lexicon, such as the German Tutor (section 2.5.2); or were limited in the number of error categories treated, such as VP^2 (section 2.7.4).

We also believe EDS to be competitive in terms of manpower needed for its creation. Reusing and adapting an existing syntactic parser is the clue to this competitiveness. A rough estimate of the person/months used for the design and implementation of EDS, as an adaptation of the Fips parser,[1] amounts to 30 person/months. We very much doubt that so few resources would have been allocated for any of the commercial grammar checkers.

Finally, our error diagnosis system is also competitive through its output which, besides error diagnosis, contains a syntactic analysis of the diagnosed sentence. This feature, which allows a CALL software to display in a user-friendly format important lexical and syntactic clues to help the users correct their productions, is seldom available in commercial grammar checkers.[2] It is therefore an important bonus in favor of our error diagnosis system.

6.1.2 Use of innovative techniques

While three error diagnosis techniques were described in chapter 3, only two were implemented in our error diagnosis system. We believe that both techniques are innovative in the ways they were employed.

The constraint relaxation technique is neither new nor rare, as can be readily seen through the wealth of literature on this technique. Nonetheless, the manner in which we used it in the error diagnosis system is innovative. The innovations concentrate on three areas: (i) the addition of constraint relaxations after completion of the grammar and/or the parser, (ii) the types of errors detected through constraint relaxation, and (iii) the number of constraints relaxed at the same time.

Firstly, most grammar checkers are built from scratch and the diagnosis techniques are designed together with the parser, if a parser is indeed included. This was naturally impossible in our case as we reused an existing

[1]That is, without counting the time and resources necessary to design and implement the Fips parser itself.

[2]'Le Correcteur 101', of Machina Sapiens, and 'Antidote 98', of Druide Informatique, are notable exceptions. We do not know, however, how fine-grained their analyses are.

syntactic parser. Thus, we had to figure out which were the constraints which interested us for error diagnosis and to add constraint relaxation mechanisms on existing constraints.

Secondly, the range of errors detected through constraint relaxation is greater and more diversified with EDS than with most other systems. Naturally, we use constraint relaxation for agreement errors, as it is rather common to do. However, we also employ the constraint relaxation technique to diagnose errors which have an influence on the syntactic tree built by the parser. In particular, we use this technique to detect word order errors, as well as verb and adjective complementation errors. This is an innovative use of the technique which is not usually utilized for this kind of errors.

Finally, we employ the constraint relaxation technique in an innovative manner through the number of error categories detected through constraint relaxation. We detect fourteen distinct error categories through this technique and, as an error category corresponds most often to several constraints, this implies that a great number of constraints are relaxed at the same time. Most error diagnosis systems using constraint relaxation seem to have fewer relaxable constraints.

Phonological reinterpretation is a much more innovative technique than constraint relaxation. Although some of its components have been used before in different systems, either for spell checking or for grammar error diagnosis, it is the first time, to our knowledge, that they are gathered together to form a coherent error diagnosis technique. The main innovation resides in trying to parse a sentence, which so far had not received a complete analysis, with alternative words which are homophones of the ones found in the original sentence. Thus, homophones of all kinds, not just the most common ones like the *a/à* (has/to) pair, can be detected. This diagnosis of homonymy errors does not resort to error lists, by necessity limited in the number of items they contain, and is therefore much more generic than what can be found in most grammar checkers.

The phonological reinterpretation technique also allows the system to detect some instances of the incorrect use of morpho-syntactic features such as tense, mode, and agreement, as long as these features are not overtly realized in the pronunciation. While agreement errors can easily be detected with other diagnosis techniques, tense and mode errors are usually not treated. Moreover, they are often the source of incorrect analyses of a sentence. Although the diagnosis of the errors detected through phonological reinterpretation has not been refined yet, these errors can be detected and corrected by our error diagnosis system. Homonymy errors are more frequent with language learners than with native speakers and are not usually treated by commercial grammar checkers. It is therefore innovative to detect them and goes well in line with the tailoring of our error diagnosis system for a target user population of language learners.

6.1.3 Adapting an existing syntactic parser

This dissertation has proven the feasibility of adapting an existing syntactic parser to transform it into a grammar error diagnosis system. Starting from an existing system has some very interesting advantages but it also has some drawbacks.

The main advantages of reusing a syntactic parser are found in the work which has already been accomplished. The diagnosis systems benefits from the coverage in terms of grammar and lexicon, from the reliability, and from the robustness of the existing parser. Before any modification is brought to the parser, it should theoretically already work as a recognizer, sorting correct sentences from ungrammatical ones. Thus, only sentences which contain errors can be submitted to diagnosis, therefore reducing processing time for grammatical sentences and eliminating the risk of producing overflags on them. Moreover, the parser underlying the diagnosis tool is able to provide syntactic analyses of the submitted sentences, which have multiple applications in a CALL software.

Specific advantages of Fips, besides its very large coverage, were its non determinism and its capacity to provide partial analyses when it is not able to find an analysis covering a whole sentence.

The major drawback of adapting an existing parser is the restrictions the parser can impose on the choice of diagnosis techniques. Indeed, not all techniques are compatible with all types of parsers. Another disadvantage with large coverage parsers is the complexity of their algorithm and code. This complexity makes it sometimes very arduous to locate the proper position for a modification or the introduction of a diagnosis technique.

Compared to what might be expected of a syntactic parser, Fips has the particular disadvantage of not being truly a recognizer. It accepts many ungrammatical sentences in order to broaden its coverage. Moreover, its lexicon is sometimes the source of overflagging or of non detection of errors because of incorrect or missing information on specific lexical features.

The balance between the advantages and the disadvantages of transforming a syntactic parser, and in particular Fips, into a grammar error diagnosis system is strongly positive. Without Fips as a basis, it would have been impossible to create a diagnosis system such as EDS, with a large coverage of the grammar and a relatively large coverage of error categories, in the time frame which was allocated to us. Only reuse of an existing parser allowed us to go beyond the toy prototype stage.

6.2 Combination of several techniques

One of the aims of this dissertation work was to combine several diagnosis techniques together in order to investigate in which manner they could be combined and how to deal with the results emanating from each different technique. We also wanted to determine whether the use of several techniques was an advantage or not over the use of only one technique.

Unfortunately, we have been able to implement, and therefore to combine, only two error diagnosis techniques compared to the three planned. This, therefore, restricted the combination possibilities available to us. Moreover, while we would have liked to investigate different manners to combine the two techniques, we were limited in time and could not experiment as much as we wanted to. Thus, only the cascade combination was tried out and implemented, phonological reinterpretation being launched only when constraint relaxation did not provide a complete analysis for a given sentence.

The combination of diagnosis techniques should imply competition between the results obtained through the different techniques. While some error categories can be detected by both constraint relaxation and phonological reinterpretation, there is in fact no competition between their respective results. Indeed, if constraint relaxation permits a full analysis of the sentence, phonological reinterpretation is not activated and therefore produces no result. When phonological reinterpretation is launched, then constraint relaxation has to be disabled for efficiency reasons, and therefore only the results emanating from phonological reinterpretation can be displayed. Thus, in the particular case, there is no competition between the results of different error diagnosis techniques and we have not had to find discriminating ways between them, which prevented us from carrying out an interesting investigation.

The two diagnosis techniques which we have combined in this error diagnosis system overlap partially in terms of error categories detected. A partial overlap also means that some error categories are distinct, that is that each technique detects errors that the other technique is not capable of detecting. Thus, by combining the two techniques, we have enlarged the range of detected errors, which is a definite advantage. In view of this, one is tempted to assume that the more diagnosis techniques, and thus the more error categories detected, the better. This is unfortunately not necessarily true, as each diagnosis technique takes its toll on efficiency. However, it is possible that if each diagnosis technique is even more specialized, its particular efficiency might be enhanced. Thus, as long as different error categories are targeted by each diagnosis technique, it seems worthwhile to try the combination of as many techniques as available.

6.3 Limitations of the work accomplished

Any work naturally has its limitations and this dissertation is no exception to the rule. Some of the limitations are due to the global research project into which the creation of the grammar error diagnosis system inscribed itself, as the goals of the FreeText project had to be taken into account during the development of the grammar checker. However, the vast majority of the limitations are due to time issues. Many improvements of the system could be achieved if more time was taken to implement them. The main ones are cited below.

More precise specifications for chunk reinterpretation than those found in section 3.3 have not been designed and the technique itself has not been implemented. This is an important part of the work initially planned which was not carried out. Besides the time issues involved, reasons for this are that the error categories which we planned to detected through chunk reinterpretation were already covered by either one of the two implemented diagnosis techniques. Thus, it is highly possible that not implementing chunk reinterpretation has not diminished the error coverage of the final error diagnosis system.

Overflagging is one of the major remaining issues of the current version of the error diagnosis system. Putting down this unaccomplishment to time reasons solely would not be true as the issue was seriously worked upon. Numerous bugs were discovered and corrected in the implementation which had for effect to reduce the number of overflagging occurrences but did not contain it sufficiently. Some *ad hoc* measures were also taken, such as displaying the results of phonological reinterpretation only when a full sentence analysis had been reached. Nevertheless, one hypothesis to reduce overflagging occurrences has not been tried because of time restrictions: the relaxation of constraints by packets. Implementing packets, as well trying and testing different ways in which to assemble these packets, would have been very interesting, but also very time consuming for an uncertain result.

The current implementation of the phonological reinterpretation technique has proven to lack in efficiency and efforts should be made to better its implementation. At present, reinterpreted words are simply added in the chart of well-formed constituents resulting from a first parse of the sentence, and the parsing process is restarted. The elements contained in the chart and created during the first pass might be cluttering the chart and forcing the system to try redundant alternative paths, thus demanding more time and resources to reach an analysis. It is highly probable that a more careful management of the chart's content before the parsing process is restarted would help improve the efficiency of phonological reinterpretation.

Some of the error categories chosen for diagnosis with EDS could benefit from a better treatment. The homonymy error category must be refined so that only true homophones are flagged under this error tag. Other errors discovered through phonological reinterpretation should receive tags appropriate to the actual error present in the text. This involves comparison procedures between the original and the reinterpreted words in order to assess whether they share the same root form and, if so, which are the distinct features which caused the error. The auxiliary and voice error categories should be redefined so as to eliminate any confusion on the appellation of errors on the passive voice. Also, one should find ways to detect the erroneous use of the *être* (be) auxiliary even with verbs which can be put in the passive. This type of error remains undetected currently. Verb complementation is another area where improvements are called for. The current treatment of preposition confusion and adjunction is not satisfactory as errors are basically ignored with nominal complements. The whole verb complementation constraint might need revision to better account for the attachment of adjuncts, instead of complements, to unsaturated verbs and to improve the management of multiple subcategorization frames for a single verb.

A complete verification of the lexicon used with the error diagnosis system is called for. More accurate and coherent lexical information can only improve the performances of EDS and diminish the number of overflagging occurrences. One should, for example, check that all prenominal adjectives do indeed bear the prenominal feature, that all lexemes of a verb which cannot passivize are marked as such, and adverb categories are correctly set for each adverb. This is a mammoth task which is outside the scope of this dissertation but which is crucial to obtain better results with EDS.

The limitations of the implemented diagnosis system account for some of the limitations in the experiments intended on the combination of diagnosis techniques and on the competition between diverging results. Only two of the three planned diagnosis techniques were finally implemented. For their combination, the cascade method seemed the best suited and was implemented without further ado, as there was not time, and less interest with only two techniques at hand, to try out other methods. Moreover, as we had to disable constraint relaxation when phonological reinterpretation was activated, there was no possibility of competing results between the two diagnosis techniques. This whole experimentation scheme had therefore to be abandoned.

6.4 Further research

Much remains to be done in the field of grammar checking in the context of computer assisted language learning. We are still far from tools which cover all error categories committed by language learners in a robust, reliable, and efficient manner. Thus, there are many paths open for further research. A few are cited below.

The language level of the diagnosis system users determines, to some extent, the error types which are likely to occur in their productions. Therefore, it might be advantageous to parameterize EDS according to language levels. The language level might change the error frequencies and, thus, the score to be used for analysis retrieval. It might also help ruling out some complex sentence structures for less advanced users. Moreover, one does not necessarily want to be as strict with beginners as with advanced learners. The introduction of parameters would allow the system to flag error categories only if they are relevant for the current language level. Redundancy errors, for example, could be reserved to the most advanced language learners.

There might be some specific errors, either rather rare but problematic for parsing, such as the missing apostrophe in example (119), page 193, or extremely frequent, like the erroneous use of the homophones *a*/*à* (has/to), which would be more appropriately treated with exceptional measures, rather than being included into a regular diagnosis technique. We could thus set up a list of errors to be treated in an exceptional manner. The treatment of these errors could be done as a preprocessing task, thus facilitating the rest of the diagnosis.

We have seen throughout this dissertation that homophones were a real difficulty for language learners and that most commercial grammar checkers did not detect errors of this kind, bar on some very frequent pairs of homophones (such as *a*/*à* (has/to)) for which *ad hoc* measures have been devised. As has been shown, homonymy errors can be detected, diagnosed and even corrected through phonological reinterpretation. However, there is another similar problem: near homophones (see example (121)). Besides the errors themselves which must be detected, their presence can be very detrimental to the whole parsing process.

(121) *qui* vs. *qu'il*
/ki/ vs. /kil/
who vs. which he

The problem of near homophones has been touched upon in section 3.2.7. However, the solutions proposed there, introducing a distance measure or permitting some specific phonological changes, seem inapplicable given the efficiency difficulties already encountered with strict phonological reinterpretation. While we could consider using lists of very frequent near homophones like example (121), this would not be feasible for less frequent, or even unique, instances of near homophone confusion, such as example (122).

(122) *parler* vs. *parlait*
/paʀle/ vs. /paʀlɛ/
talk vs. talked

For a general treatment of near homophones, the implementation of phonological reinterpretation has to be greatly improved, or a completely different diagnosis technique has to be thought of.

More or less fixed phrases (compound words, idioms, and collocations) is a domain which even very advanced learners find difficult to master. Their proper use is however necessary to achieve a good level of proficiency. Detecting errors in the use of fixed phrases, either an incorrect word order as in example (94), page 174, or use of an incorrect word in an expression such as in (123) where the appropriate verb is *battre* (beat), would be a great help to advanced learners who are unlikely to find help in this matter with grammar checkers designed for native speakers.

(123) **Fouetter les œufs en neige.*
Whip the eggs in snow.

How to diagnose errors in fixed phrases is a completely new domain to be investigated, as these errors are not really syntactic in nature. Solutions to this problem are bound to be very dependent of the way in which the fixed phrases are encoded in the lexicon and treated by the parser.

Another domain in which research should be conducted in order to improve the state of grammar checkers for language learners is diacritics. As we have seen in section 5.2.2, errors on diacritics which are not caught by a spell checker because they result in legal words raise many difficulties for parsing and thus for error diagnosis. Diacritic errors are particularly visible on past participles where they might prevent the detection of other errors, linked to the difficulties of past participle agreement in French. Investigations might be pursued along the lines of Simard and Deslauriers (2001) who propose a statistical method for re-inserting accents in a text. This method would have to be adapted to language learner productions, as well as to incorrect diacritics along with missing ones.

However, if missing diacritics are mostly problematic on past participles, another option would be to focus detection not on diacritics, but on the past participles themselves. The method could be in charge of locating erroneous past participles, detecting errors of diacritics, tense and/or mode, choice of auxiliary, as well as agreement errors. This would be a shift in orientation from focusing on an error category to concentrating one's efforts on a problematic lexical category or syntactic construction.

This shift in focus would connect together lexical, morphological, and syntactic errors. We have, in this dissertation work, taken the simplified view of separate levels of checking, independent from one another. Moreover, we have concentrated solely on the grammatical aspects, dabbing into morpho-syntax only when necessary. Much could be gained, however, by the integration of these different levels into a single error diagnosis system which would take into account all levels of checking before offering a diagnosis. Spell checking could greatly improve its correction proposals by taking into account the morphological and syntactic contexts in which erroneous words are located. Morphological errors, which are often treated by neither spell nor grammar checkers, apart from signaling some of them as unknown words, could be integrated into such a system. They would thus be detected even when the resulting words exist, and they would receive a diagnosis coherent with their syntactic environment. Finally, the syntactic level of checking would largely benefit from parsing with more correct sentences at the lexical and morphological levels, thus improving syntactic error detection and diagnosis.

Further developments, once the lexical, morphological, and syntactic levels of checking are well in place, are the expansion towards semantic and pragmatic levels. In parallel, adding correction proposals to this multilevel checker would complete this tool and transform it into a comprehensive correction system.

6.5 Final remark

Language learners have specific needs which are not catered to by commercial grammar checkers built for native speakers. We have tried to answer some of the language learners' needs through the design and implementation of a grammar checker specifically designed to treat some of the error categories often committed by them. This error diagnosis system can be usefully integrated into a larger computer assisted language learning software for the diagnosis of free production exercises, thus providing a real-time intelligent feedback to the language learners and allowing CALL software designers to create more challenging exercises.

Appendix A

List of Abbreviations

This appendix contains a list of abbreviations which can be found throughout this dissertation. For some synonymous abbreviations, the author has her own preferences which do not always tally with examples found in cited work.

ACFG	Augmented Context Free Grammar
ALLP	Athena Language Learning Project
ARCTA	Aide à la Rédaction et à la Correction de Textes Anglais
ARI	U.S. Army Research Institute
ASL	American Sign Language
ATN	Augmented Transition Network
AUX	Auxiliary error category
BAP	BAsic Parser
BKSW	Black, Kwasny, Sondheimer and Weischedel's family of systems
CALI	Computer Assisted Language Instruction
CALICO	Computer Assisted Language Instruction Consortium
CALL	Computer Assisted Language Learning
CECL	Centre for English Corpus Linguistics, *Université Catholique de Louvain*, Belgium
CLA	Class error category
CPA	Adjective complementation error category

CPV Verb complementation error category

EAES Error Analysis and Explanation System

EDS Error Diagnosis System

EFL English as a Foreign Language

ELSE Elementary Language Study Experiment/Exerciser

ESL English as a Second Language

EUF Euphony error category

FRIDA FRench Interlanguage DAtabase

French RObust Grammar analyser

FSA Finite State Automata

FSGC Francophone Stylistic Grammar Checker

GEN Gender error category

GB Government and Binding

GPSG Generalized Phrase Structure Grammar

HOM Homonymy error category

HPSG Head-driven Phrase Structure Grammar

ICALI Intelligent Computer Assisted Language Instruction

ICALL Intelligent Computer Assisted Language Learning

ICICLE Interactive Computer Identification and Correction of Language Errors

ILTS Intelligent Language Tutoring System

ISCA Interactive Sentence Constructor and Analyser

ITS Intelligent Tutoring System

L1 First language

L2 Second or foreign language

LFG Lexical Functional Grammar

LINGER Language INdependant Grammatical Error Reporter

MAN Missing element error category

MS Menzel and Schröder's system

NBR Number error category

NEG Negation error category

NLP Natural Language Processing

NLU Natural Language Understanding

ORD Order error category

ORDAJ Adjective order error category

ORDAV Adverb order error category

OUB Missing punctuation error category

PER Person error category

POS Part Of Speech

RECALL Repairing Errors in Computer Aided Language

RED Redundancy error category

SLA Second Language Acquisition

TESOL Teachers of English to Speakers of Other Languages

UCL *Université Catholique de Louvain*, Belgium

UMIST University of Manchester Institute of Science and Technology, United Kingdom

VOI Voice error category

XML eXtended Mark-up Language

Appendix B

Sample test corpora

This appendix contains sample corpora in order to exemplify the data on which the error diagnosis system was tested and from which results for chapter 5 were derived. Both corpora of 'linguist' and 'simplified' sentences contain errors of the verb complementation (CPV) category. The 'authentic' sentences, taken from the FRIDA corpus, contain euphony (EUF) errors (among other). Finally, the 'authentic' sentences coming from a validation phase of the FreeText software, the so-called FreeText data, do not represent any specific error category.

B.1 'Linguist' sentences

Je dors le chien.
Je lui dors.
Je bois au vin.
Je lui bois.
Le vin à qui je bois.
J'y bois.
Le cadeau que je l'offre.
L'ami à qui je lui offre.
Je lui offre à un ami.
J'offre le cadeau sur un ami.
Je ressemble de cette personne.
Je ressemble cette personne.
L'ami duquel je ressemble.
Je lui ressemble un ami.
J'espère à venir.
J'espère de venir.
Je l'espère venir.
J'évite à venir.
J'évite qu'il part.

Je l'évite de partir.

B.2 Simplified sentences

Tout le monde a le droit d'aller l'université.
Cela bénéficiera la Grande-Bretagne.
Il est inutile de débattre l'effet de cette déclaration.
On commence à discuter les détails.
Il faut discuter ce sujet.
On ne devrait pas douter la question.
On ne peut pas échapper notre nature.
Elles ont échappé l'opinion publique.
C'est ce que je peux fournir l'acheteur.
L'Europe goûte les bénéfices.
Elle jouissait un grand rôle.
Les Anglais se méfient tout.
Elle offre les Britanniques une chance.
Cela va plaire tout le monde.
Elle se rallie la loi constitutionnelle.
Elle doit renoncer sa souveraineté.
Le gouvernement sera obligé de renoncer un certain degré de contrôle.
Les filles souffrent un entraînement.
Cela aidera à augmenter le respect de soi et donc être plus content.
Il choisit acheter un bonbon.
Un grand nombre de personnes craignent perdre leurs propres coutumes.
Il s'est engagé travailler.
On peut parler de la vie et nos expériences.
Ils s'accordent à beaucoup de choses.
C'est un problème associé avec la souveraineté.
Elle veut concourir dans le marché européen.
Il se conforme avec les normes.
L'art consiste de plusieurs mondes.
Nous n'avons rien à craindre avec l'Europe.
Il croit à Dieu.
Chaque étage dépend sur l'autre.
Cela la différencie aux autres.
Cela donne une stabilité dans la société.
La souveraineté pourrait échapper de la Grande-Bretagne.
Il fallait enseigner des hommes.
Elle excelle à son métier.
Il fournit des munitions pour les ennemis.
Il s'intéresse du chômage.

Il s'inquiète à cette idée.
Il se joint avec l'Europe.
Il se mêle avec les autres pays.
Il s'oppose contre leur désir.
Il s'oppose de leur désir.
Participer dans une guerre sainte.
Je peux me passer sans elle.
Je pense d'eux.
Je les prépare pour l'examen.
Je profite au beau temps.
Cela se rapporte pour les activités.
Je me rapproche à la route.
Cela se réduit en rien.
Je résiste des problèmes.
La bourse a aidé d'intégrer les travailleurs.
On cherche de continuer la recherche.
Il consent de partager.
Il décide à joindre le parti.
Il se délecte à ne pas être comme les autres.
Cela les a empêchés à ruiner l'environnement.
Il l'entraîne de marcher.
Ce qu'il est permis à faire.
Il a réussi de monter le meuble.
Elle tente à devenir une championne.
On ne peut pas aider à tous les pays.
Nous concurrencerons avec les autres membres.
Nous combattrons à l'ennemi.
Je connais sur nos voisins.
On nous fournit avec les moyens nécessaires.
Elle veut maximiser de ces chances.
Elle partagera dans les bénéfices.
Elle promeut de la liberté.
Elle rencontre avec des gens.
Ils doivent soigner de leur famille.
Chacun doit de ne pas dépasser la limite.
Elle espère de régler le problème.
Il paraît à supporter l'écu.
Chacun peut d'abuser de la situation.
Je préfère de partir.
Cela semble d'être vrai.
Il vaut mieux de se joindre à eux.
Elles voulaient d'être respectées.

B.3 Authentic sentences

B.3.1 FRIDA data

On veut de nous de les suivre, de faire comme eux “seulement pour notre réussite” mais en cachant de nous tous les sentiments que éprouvent les êtres humains.
En prenant ce aspect il ne peut pas y avoir des avantages ou d’inconvénients d’être une femme ou un homme.
Y-a-il un avantage d’être un homme? Hier soir, le vingt mai 1992, un billion d’amateurs du football a regardé le match à Wembley entre deux équipes européennes, de un côté il y avait une équipe espagnole et de l’autre coté une équipe italienne.
Plusieurs incidents de l’hoolganisme ont résulté avec des morts des amateurs.
Tandis qu’il y existe des aspects mauvais du sport déjà mentionnés: l’hooliganisme et des frais des rassemblements des gens sportifs, on ne peut pas nier des avantages dont les plus importantes inclurent l’amélioration des relations diplomatiques et internationales.
Par exemple, sous le reine d’Hitler, la peinture devenait très conservatrice et très propagandiste.
L’art consiste de plusieurs mondes il ne faut pas qu’il ressemble à le nôtre.
Pendant qu’elle est bonne, en tant que telle, elle a permis à les femmes aussi de se coucher à droit et à gauche; elle a apporté une moralité relâchée.
Souvent, donc des femmes se trouvent trompées, spirituellement vidées.
Peut-être en ce contexte on comprend l’avis de Simone de Beauvoir qu’ “on ne naît pas femme, on le devient”.
Les deux côtés du argument peuvent être justifié.
Jusqu’à présent sous madame Thatcher le Royaume Uni avait resté favorablement hostile à les moindres danger comme la création d’une banque centrale européenne et d’une monnaie unique.
Si les anglaises ne utilisent pas sa chance unique pour qu’ils deviennent plus indépendantes, la Grande Bretagne deviendra un pays faible par rapport tous ses compétiteurs comme l’Allemagne ou la France par exemple.
Le nouvel marché aidera les 12 car la disparition des tarifs restrictifs encouragera l’industrie.
L’idée d’un Europe unifié peut être utile pour tout le monde mais je pense que à une certaine mesure chaque pays doit se démarquer et garder sa identité pendant qu’ils s’efforcent d’améliorer les relations et de former une unité beaucoup plus cohésive.
Nous avons déjà devenu partie de la CEE(en 1992) qui a été former afin de unifier les bénéfices économiques de chaque pays.
La Grande-Bretagne a peur de confondre son identité avec celui de l’Europe et cela explique la hostilité des britanniques contre une monnaie européenne

unique.
Mais c'est un peu trop tard pour se plaindre parce que il a été accepter dans la Traité de Rome que le loi fait par le parlement Européen a la préséance sur le loi des pays qui fait partie de la Communauté.
L'économie, ainsi, en Grande Bretagne va améliorer; néanmoins les Britanniques sont tout à fait contre un monnaie Européen unique, l'ECU, parce que un monnaie unique est une perte de leur identité et leur indépendance.
Donc à cause de sa histoire Grande Bretagne est devenu assez différente et indépendante en comparaison avec les autres pays d'Europe.
Mais encore une fois il faut dire que un des défauts des CEGEPs, c' est le grand choix qu'on a.
Mais, c'était ce qu'on appelle aujourd'hui "la fausse régionalisation", parce ce que elle n'a rien changé pour aider des habitants.
Donc, il y a une différence entre les deux livres de l'attitude envers la société, bien que il est vrai que les deux acceptent les autres: les Chapdelaines n'ont pas envie de la vie urbaine(même à la fin Maria,) et Galerneau devait aller en ville.
Parce que ils sont différents des Anglais.
C'est parce que les continentaux sont plus différents aux anglais.
En 1960, avec son "équipe du tonnerre" Monsieur Nésage a proclamé que "il est temps que ça change".

B.3.2 FreeText data

A l'âge de 7 ou 8 ans je n'avais pas réfléchi au sujet de la mortalité et cela m'a cause un choc.
Au début je ne pouvais pas supporter la solitude de ma chambre, jusqu'au moment où j'ai vu par ma fenêtre un groupe de oiseaux qui volaient.
Aujourd'hui tout va bien c'est presque comme si cet événement n'a jamais eu lieu.
Ce jour-là, il faisait très beau et on faisait la file pour deux heures.
Celui où j'ai me suis rendu compte de leur futilité et de leur humanité comme un enfant sur lequel tombe soudainement l'idée affreuse: ses parents ne sont pas des héros tous puissants.
C'est la même chose quand je suis dans la voiture avec quelqu'un et quelque chose se passe.
C'était la semaine avant Noël, j'avais seulement 12 ans.
C'était merveilleux! Elle avait beaucoup de choses de très grande beauté et ils étaient toutes pour moi ce matin-là.
C'était un jour comme tout les autres.
C'était une conversation intéressant.
Comme le narrateur du texte, je suis devenue grande ce jour-là.
Depuis ce jour j'ai décidé que je dois devenue une actrice.

Depuis ce jour je n'ai pas pu aimer un chien pareil.
Depuis ce jour je ne roule plus.
Depuis ce jour je n'y ai plus cru.
Depuis ce jour je regarde le rouge et les voitures avec beaucoup d'attention.
Depuis ce jour là je n'ai plus regretté ma décision et j'ai exigé de plus en plus la valeur de la liberté.
Depuis ce jour la, je n'ai plus envie de voler.
Depuis ce jour là, je ne peux pas s'arrête de lui penser.
Depuis ce jour la, je prend un gilet avant de me mettre a l'eau.
Depuis ce jour ma vie a été renouvelée.
Depuis ce jour mes parents son divorcé.
Depuis ce jour nous habitions une grande maison.
Depuis ce jour si je suis chez moi et il fait un temps semblable à ce jour-là - il pleut à verse - et mon père est en retard je commence à me demander où il est.
Depuis ce jour, je l'écris et nous sommes encore des amies.
Depuis ce jour, je ne mange plus la crème glacée.
Depuis ce jour, j'espère devenir une poète encore.
Depuis ce jour, le goût des parcs d'attraction a fortement diminué.
depuis ce jour, ma vie a changé.
deux jours après elle est morte.
Elle a vécu pour deux années.
Elle est très vieille mais très contente parce que tous les enfants (moi, mon frère et autre cousins ou cousines) lesquelles elle a soigné sont grandi est ont la vie heureuse.
elle habitait dans le campagne, et le samedi nous rendions visite à ses amis.
Elle m'avait accompagné jusqu'à j'avais 6 ans.
Elle me bien soignait: elle faisait toujours le bon nourriture est me donnait à manger; elle me racontait les histoires intéressants le soir.
Elle s'est appelé Suey et elle était un cadeau pour mon troisième anniversaire.
Elle venait de Paraguay mais elle vivait en Belgique.
Et je rêvait comme rêvent les enfants qui jouent dans la rue à soldats et a cow-boys.
Finalement, même si je l'aimais beaucoup, j'ai ouvert la petite porte de sa cage et je l'ai délivré.
Heureusement elle a réussit à le trouver.
Il avait l'air tellement malheureux que j'ai commencé réfléchir sur la liberté et la chance que j'avais.
Il était vivant mais gravement blessé.
Il voulait parlait de mon père.
Il y avait un monsieur du coin.
Ils étaient mes héros.
J'ai mis un beau pantalon et des chausseurs rouges.

J'ai pris des leçons et depuis ce jour là, je joue plusieurs instruments.
J'arrivais de l'école et je me mettais à étudier l'histoire de mon pays et tous les présidents qu'il avait eu.
Je étais très contente de vivre avec ma grand mère.
je l'aimerais parce qu'elle était très gentille, et chaque jours elle me disait des histoires très drôles.
je me sentais déçu est fâché contre moi même pour avoir tombé dans leurs nets de mensonges, d'illusion, illusion enfantine qui donne l'impression d'une pouvoir épouvantable; selon laquelle tu peux changer le monde.
Je me souviens je suis très triste dans le jour où elle est partie; j'ai pleut sans arrêt et j'ai refusé de rentrer à la maison.
Je ne suis plus une petite fille insouciante de tout.
Je pensais sur leurs actions: pourquoi tel avait fait telle chose, pourquoi tel autre n'avait pas fait telle autre...
Je suis restée en dehors la maison jusqu'à le soleil est disparu et les vedette sont sorti.
La musique est très importante dans ma vie.
Le professeur de musique était très sympathique et il inspirait l'enthousiasme parmi la plupart des enfants.
Lui, dans son camion a presque écrasé la camionnette de mon père.
Ma grande mère me regardait et demandait pourquoi je suis devenue tout pale.
Ma mère a ouvert la porte.
Ma mère était beaucoup en colère et elle n'avait jamais me permis d'encore entrer dans sa armoire.
Ma mère nous a racontait qu'il parlait comme si mon père était mort.
Maintenant je grandir et deviens un adulte.
Mais l'événement m'a vraiment marqué.
Mais soudainement j'ai tombe et après ca, je devais aller a l'hôpital.
mais un jour elle disait qu'elle avait mal à la tête.
Mais un jour elle est partie à l'autre ville loin pour soigner mon cousin (le fis de mon ongle).
Mais un jour elle est repartie pour Paraguay.
Mais un jour J'ai eu le chance a parler avec le directeur de la filme.
Mais un jour j'ai mange la crème glacée empoisonner.
Mais un jour j'ai reçue un mal essai de l'anglais.
Mais un jour j'ai trouvé que j'étais trop grand pour la porter.
Mais un jour je suis allé trop loin dans le lac et j ai fait noyer.
Mais un jour je suis tombée.
Mais un jour mon chien est mort.
Mais un jour mon père a gagne beaucoup d'argent dans un concours d'échec.
Mais un jour tout a change.
Mais un jour tout comme le petit garçon dans l'histoire un grand événement a eu lieu.

Mais un jour une voiture est arrivé trot vite, et je suis tombé par terre.
Mais un jour, après une longue recherche, mes parents ont trouvé une nouvelle maison, j'avais une chambre pour moi, il y avait un jardin et ensuite on a pu prendre un chien.
Mais un jour, ma mère a nous prend à une autre payé sans mon père.
mais un jour, mon poisson rouge est mort.
Mais, soudainement quelqu'un a sonné à la porte.
Moi c'était le même avec les politiciens; je les voyais comme corrompus et menteurs quand avant ils avions été des dieux.
Moi, assise dans la chambre d'à côté, j'entendais aucun de ses mots mais de la façon dont ils parlaient je savais que quelque chose d'horrible s'est passé.
Moi, je l'adorais mais un jour j'ai remarqué que il regardait tout le temps dehors de ma fenêtre.
Mon grand mère est encore vivant.
Quand j étais petite j aimais nager.
Quand j'étais petite j'ai eu une robe rouge qui étais ma robe favorite.
Quand j'avais sept ans, ma grand-mère m'a acheté ma première violon.
Quand j'étais petit j'avais envie de voler comme un oiseau mais un jour en jouant avec mes amis, quelqu'un a tombée du toit.
Quand j'étais petit, ce que je voulais le plus, c'était devenir politicien.
Quand j'étais petit, mon chien était un de mes meilleurs amis.
Quand j'étais petite.
Quand j'étais petite j'ai beaucoup aime la crème glacée.
Quand j'étais petite j'ai pensé que ma famille était très content.
Quand j'étais petite j'ai reçu un vélo.
quand j'étais petite j'aimais beaucoup les animaux mais un jour j'ai vu un chien qui a tue une bébé et depuis ce jour j'ai déteste les animaux.
Quand j'étais petite j'avais un petit oiseau dans une cage.
Quand j'étais petite je n'avait aucune souci.
Quand j'étais petite je toujours promenais sans aucun attention, j'étais une fille très impatient, et je toujours croissait le rouge follement.
Quand j'étais petite ma mère et moi sommes allées au cinéma.
Quand j'étais petite, j'adorais aller souvent à un parc d'attraction.
Quand j'étais petite, j'adorais la musique.
quand j'étais petite, j'ai vu un homme avec une moustache qui était plus long que ses cheveux.
Quand j'étais petite, j'aimais bien les leçons de musique a l'école.
Quand j'étais petite, j'avais une amie.
Quand j'étais petite, je vivais avec ma grand mère.
Quand j'étais petite, j'espère devenir une poète.
Quand j'étais petite, j'était très amusant un jour parce que ma mère a sorti et je pouvais essayer toutes ses vêtements de son armoire.
quand j'étais petite, j'habitais avec ma grand-mère.
Quand j'étais petite, ma famille et moi habitions dans un grand immeuble

à la périphérie de la ville.
Quand j'étais petite, ma famille était pauvre.
Quand j'étais une petite, j'étais toujours contente.
Quel désastre! Depuis ce jour je n'ai plus avoir une robe favorite parce que je n'aime pas le chagrin quand je deviendra trop grand! Ton voix, c'est ton voix, c'est un trésor Mon voix peut changer leurs vies, mais pas la mienne.
Tout de suite elle s'est mis à téléphoner les hôpitaux pour savoir si mon père était là.
Tout de suite me mère est entrée dans la salle et tout devenait clair, le monsieur qui était déjà parti parlait d'un accident routier.
Un jour, mes parents m'ont prise au Wallibi.
Un jour, pourtant, il est mort d'une crise cardiaque.

Bibliography

Akmajian, A., R. A. Demers, and R. M. Harnish (1984). *Linguistics: An Introduction to Language and Communication* (2nd ed.). Cambridge, Mass.: MIT Press.

Allen, J. R. (1997). Ten desiderata for computer-assisted language learning programs: The example of ELSE. *Computers and the Humanities 30*, 441–455.

Allen, R. E. (Ed.) (1990). *The Concise Oxford Dictionary of Current English* (8th ed.). Oxford: Oxford University Press.

Bailin, A. (1988). Artificial intelligence and computer-assisted language instruction: A perspective. *CALICO Journal 5*(3), 25–45.

Bailin, A. (1990). CALL, artificial intelligence and the representation of social roles. In Craven, Sinyor, and Paramskas (1990), pp. 173–179.

Bailin, A. (1991). ICALI research: A special issue of the CALICO journal. *CALICO Journal 9*(1).

Bailin, A. and L. Levin (1989). Introduction: Intelligent computer-assited language instruction. *Computers and the Humanities 23*, 3–11.

Barchan, J., B. Woodmansee, and M. Yazdani (1986). A Prolog-based tool for French grammar analysis. *Instructional Science 15*, 21–48.

Boë, L.-J. and J.-P. Tubach (1992). *De A à Zut: Dictionnaire Phonétique du français parlé*. Grenoble: ELLUG.

Bolt, P. and M. Yazdani (1998). The evolution of a grammar-checking program: LINGER to ISCA. *Computer Assisted Language Learning 11*(1), 55–112.

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

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

Brock, M. N. (1990). Customizing a computerized text analyzer for ESL writers: Cost versus gain. *CALICO Journal 8*(2), 51–60.

Buchholz, E. (1992). Factors influencing the acceptance of CALLware. *Literary and Linguistic Computing 7*(2), 132–137.

Burston, J. (1998). Antidote 98. *CALICO Journal 16*(2), 197–212.

Cameron, K. (Ed.) (1999). *CALL: Media, Design and Applications.* Lisse: Swets and Zeitlinger.

Catt, M. and G. Hirst (1990). An intelligent cali system for grammatical error diagnosis. *Computer Assisted Language Learning 3*, 3–26.

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

Chanier, T. (Ed.) (1999). *EuroCALL'99: Conference Handbook*, Besançon, France. EuroCALL.

Chanier, T., M. Pengelly, M. Twindale, and J. Self (1992). Conceptual modelling in error analysis in computer-assisted language learning systems. In Swartz and Yazdani (1992), pp. 125–150.

Chapelle, C. (1989). Using intelligent computer-assisted language learning. *Computers and the Humanities 23*, 59–70.

Chapelle, C. A. (2001). *Computer Applications in Second Language Acquisition.* The Cambridge Applied Linguistics Series. Cambridge: Cambridge University Press.

Charniak, E. (1983). A parser with something for everyone. In M. King (Ed.), *Parsing Natural Language*, Chapter 7, pp. 117–149. London: Academic Press.

Chen, L. and L. K. Barry (1989). XTRA-TE: Using natural language processing software to develop an its for language learning. In *Proceedings of the 4th International Conference on Artificial Intelligence and Education*, Amsterdam, pp. 54–63.

Chen, S.-Q. and L. Xu (1990). Grammar-Debugger: A parser for Chinese EFL learners. *CALICO Journal 8*(2), 63–75.

Chollet, G. (1994). Automatic speech and speaker recognition: Overview, current issues and perspectives. In E. Keller (Ed.), *Fundamentals of Speech Synthesis and Speech Recognition: Basic Concepts, State-of-the-Art and Future Challenges*, Chapter 7, pp. 129–147. Chichester: John Wiley & Sons.

Chomsky, N. (1964). Degrees of grammaticalness. In J. A. Fodor and J. J. Katz (Eds.), *The Structure of Language: Readings in the Philosophy of Language*, pp. 384–389. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Cambridge, Mass.: MIT Press.

Chomsky, N. (1981). *Lectures on government and binding.* Dordrecht: Foris.

Cook, V. (1988). Designing a BASIC parser for CALL. *CALICO Journal 6*(1), 50–67.

Cornu, E. (1994). Automatic correction of French prose written by English native speakers: An LFG approach. *TRANEL 21*, 181–194.

Cornu, E. (1997). *Correction automatique des erreurs morphologiques et syntaxiques produites à l'écrit en langue seconde.* Ph. D. thesis, Université de Neuchâtel, Manuscript.

Cornu, E., N. Kübler, F. Bodmer, F. Grosjean, L. Grosjean, N. Léwy, C. Tschichold, and C. Tschumi (1996). Prototype of a second language writing tool for French speakers writing in English. *Natural Language Engineering 2*(3), 211–228.

Courtin, J., D. Dujardin, I. Kowarski, D. Genthial, and V. L. de Lima (1991). Towardds a complete detection/correction system. In *Proceedings of the International Conference on Current Issues in Computational Linguistics*, Penang, Malaysia, pp. 158–173.

Craven, M.-L., R. Sinyor, and D. Paramskas (Eds.) (1990). *CALL: Papers and Reports.* La Jolla, CA: Athelstan Publications.

Criswell, E., H. Byrnes, and G. Pfister (1992). Intelligent automated strategies of teaching foreign language in context. In Swartz and Yazdani (1992), pp. 307–319.

Davies, G. and Y. H. Wei (1997). Do grammar checkers work? a report on research into the effectiveness of Grammatik V based on samples of authentic essays by EFL students. In J. Kohn, B. Rüschoff, and D. Wolff (Eds.), *New Horizons in CALL: Proceedings of EUROCALL 96*, Szonbathely, Hungary, pp. 169–188.

DeSmedt, W. H. (1995). Herr Komissar: An ICALL conversation simulator for intermediate German. In Holland, Kapland, and Sams (1995), pp. 371–381.

Dinnematin, S. and D. Sanz (1990). Sept correcteurs pour l'orthographe et la grammaire. *Science & Vie Micro 78*, 118–130.

Eglowstein, H. (1991). Can a grammar and style checker improve your writing? *BYTE 16*(8), 238–242.

Ellis, R. (1997). *Second Language Acquisition.* Oxford Introduction to Language Study. Oxford: Oxford University Press.

Erbach, G. (1990). Syntactic processing of unknown words. In Jorrand and Sgurev (1990), pp. 371–381.

Felshin, S. (1995). The Athena language learning project NLP system: A multilingual system for conversation-based language learning. In Holland, Kapland, and Sams (1995), Chapter 14, pp. 257–272.

Feuerman, K., C. Marshall, D. Newman, and M. Rypa (1987). The CALLE project. *CALICO Journal 4*, 25–34.

Freedman, A. (1989). *The Computer Glossary: The Complete Illustrated Desk Reference* (4 ed.). New York: AMACOM, American Management Association.

Gamper, J. and J. Knapp (2002). A review of intelligent CALL systems. *Computer Assisted Language Learning 15*(4), 329–342.

Gaudinat, A. and J.-P. Goldman (1998). Le système de synthèse FIPSVox: Syntaxe, phonétisation et prosodie. In *Proceedings of the XIIème Journées d'Etudes sur la Parole*, Martigny, Switzerland, pp. 139–142.

Granger, R. H. (1983). The NOMAD system: Expectation-based detection and correction of errors during understanding of syntactically and semantically ill-formed text. *American Journal of Computational Linguistics 9*(3–4), 188–196.

Granger, S. and F. Meunier (1984). Towards a grammar checker for learners of English. In U. Fries, G. Tottie, and P. Schneider (Eds.), *Creating and Using English Language Corpora*, Language and Computers: studies in Practical Linguistics, pp. 79–91. Amsterdam: Rodopi.

Granger, S., F. Meunier, N. Verhulst, and P. Watrin (2001). Logiciels de correction automatique du français et corpus de FLE. Technical report, Centre for English Corpus Linguistics, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.

Granger, S., A. Vandeventer, and M.-J. Hamel (2001). Analyse de corpus d'apprenants pour l'ELAO basé sur le TAL. *TAL: Traitement automatique des langues 42*(2), 609–621.

Grishman, R. (1986). *Computational linguistics: An indtroduction.* Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Guberman, S. (1990). Quo vadis? software for a meaningful second language pedagogy. In Craven, Sinyor, and Paramskas (1990), pp. 31–39.

Güvenir, H. A. (1992). Drill and practice for Turkish grammar. In Swartz and Yazdani (1992), pp. 275–291.

Haegeman, L. (1991). *Introduction to Government & Binding Theory.* Oxford: Blackwell.

Harriehausen-Mühlbauer, B. (1991). The computer as a 'teacher' for grammar and style errors. *Literary and Linguistic Computing 6*(4), 269–273.

Hayes, P. J. and G. V. Mouradian (1981). Flexible parsing. *American Journal of Computational Linguistics 7*(4), 232–242.

Heidorn, G. (1993). Experience with an easily computed metric for ranking alternative parses. In Jensen, Heidorn, and Richardson (1993), Chapter 4, pp. 47–52.

Heidorn, G. E. (2000). Intelligent writing assistance. In R. Dale, H. Moisl, and H. Somers (Eds.), *Handbook of Natural Language Processing*, Chapter 8, pp. 181–207. New York: Marcel Dekker.

Heidorn, G. E., K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow (1982). The EPISTLE text-critiquing system. *IBM Systems Journal 21*(3), 305–327.

Heinecke, J., J. Kunze, W. Menzel, and I. Schröder (1998). Eliminative parsing with grader constraints. In *Proceedings of Coling-ACL'98*, Volume 1, Montreal, Canada, pp. 526–530.

Holan, T., V. Kuboň, and M. Plátek (1997). A prototype of a grammar checker for czech. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, pp. 147–154. ACL.

Holland, V. M. (1994). Lessons learned in designing intelligent CALL: Managing communication across disciplines. *Computer Assisted Language Learning 7*(3), 227–256.

Holland, V. M., J. D. Kapland, and M. R. Sams (Eds.) (1995). *Inteligent Language Tutors: Theory Shaping Technology.* Mahwah, NJ: Lawrence Erlbaum Associates.

Holland, V. M., R. Maisano, C. Alderks, and J. Martin (1993). Parsers in tutors: What are they good for? *CALICO Journal 11*(1), 28–46.

Holmes, G. (1990). Canadian computer-assisted language learning: Will it survive? In Craven, Sinyor, and Paramskas (1990), pp. 1–8.

Holmes, G. and M. Kidd (1980). The evolving case or computers in the study of modern languages. *ALLC Journal 1*, 7–10.

Hu, Q., J. Hopkins, and M. Phinney (1998). NativeEnglish^{TM} writing assistant — a CALL product for English reading and writing. In Jager, Nerbonne, and van Essen (1998), pp. 95–100.

Hubbard, P. (1992). A methodological framework for CALL courseware development. In M. C. Pennington and V. Stevens (Eds.), *Computers in Applied Linguistics: An International Perspective*, pp. 39–65. Clevedon: Multilingual matters.

Hull, G., C. Ball, J. L. Fox, L. Levin, and D. McCutchen (1987). Computer detection of errors in natural language texts: Some research on pattern-matching. *Computers and the Humanities 21*, 103–118.

Imlah, W. G. and J. B. H. du Boulay (1985). Robust natural language parsing in computer-assisted language instruction. *System 13*(2), 137–147.

Jacobs, G. and C. Rodgers (1999). Treacherous allies: Foreign language grammar checkers. *CALICO Journal 16*(4), 509–529.

Jager, S., J. A. Nerbonne, and A. J. van Essen (Eds.) (1998). *Language Teaching and Language Technology.* Lisse: Swets and Zeitlinger.

Jensen, K., G. Heidorn, L. Miller, and Y. Ravin (1993). Parse fitting and prose fixing. In Jensen, Heidorn, and Richardson (1993), Chapter 5, pp. 53–64.

Jensen, K., G. E. Heidorn, L. A. Miller, and Y. Ravin (1983). Parse fitting and prose fixing: Getting a hold on ill-formedness. *American Journal of Computational Linguistics 9*(3–4), 147–160.

Jensen, K., G. E. Heidorn, and S. D. Richardson (Eds.) (1993). *Natural Language Processing: The PLNLP Approach.* Boston: Kluwer Academic Publishers.

Jorrand, P. and V. Sgurev (Eds.) (1990). *Artificial Intelligence IV: Methodology, Systems, Applications.* Amsterdam: Elsevier Science Publishers B.V.

Kaplan, J. D. and V. M. Holland (1995). Advanced technologies for language learning: The BRIDGE project within the ARI language tutoring program. In Holland, Kapland, and Sams (1995), pp. 273–287.

Kempen, G. (1992). Language technology and language instruction: Computational diagnosis of word level errors. In Swartz and Yazdani (1992), pp. 191–198.

Krashen, S. D. (1984). *Principles and Practice in Second Language Acquisition.* Language Teaching Methodology. Oxford: Pergamon Press.

Kreyer, S. and E. Criswell (1995). Instructor as author in an adaptive, multimedia foreign language tutor. In Holland, Kapland, and Sams (1995), pp. 45–54.

Kübler, N. and E. Cornu (1994). Using automata to detect and correct errors in the written English of French-speakers. *TRANEL 21*, 235–245.

Kwasny, S. C. and N. K. Sondheimer (1981). Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems. *American Journal of Computational Linguistics 7*(2), 99–108.

Labrie, G. and L. P. S. Singh (1991). Parsing, error diagnostics and instruction in a French tutor. *CALICO Journal 9*(1), 9–25.

Laenzlinger, C. (1998a). *Comparative Studies in Word Order Variation: Adverbs, pronouns, and clause structure in Romance and Germanic.* Amsterdam: John Benjamins.

Laenzlinger, C. (1998b). Les outils de TALN du LATL sur Internet. *Langues 1*(1), 82–85.

Laenzlinger, C. and E. Wehrli (1991). Fips: Un analyseur interactif pour le français. *T.A. Informations 2*, 35–49.

Letellier, S. and J.-P. Fournier (1990). How to deal intelligently with the unexpected? In Jorrand and Sgurev (1990), pp. 393–402.

Levin, L. S. and D. A. Evans (1995). Alice-chan: A case study in icall theory and practice. In Holland, Kapland, and Sams (1995), Chapter 5, pp. 77–97.

Levin, L. S., D. A. Evans, and D. M. Gates (1991). The ALICE system: A workbench for learning and using language. *CALICO Journal 9*(1), 27–56.

Levison, M., G. Lessard, and D. Walker (2000). A multi-level approach to the detection of second language learners errors. *Literary and Linguistic Computing 15*(3), 313–322.

Liou, H.-C. (1991). Development of an English grammar checker: A progress report. *CALICO Journal 9*(1), 57–70.

Loritz, D. (1992). Generalized transition network parsing for language study: The GPARS system for English, Russian, Japanese and Chinese. *CALICO Journal 10*(1), 5–22.

Manning, C. D. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, Mass.: The MIT Press.

Marcus, M. P. (1980). *A Theory of Syntactic Recognition for Natural Language.* Cambridge: MIT Press.

Marcus, M. P. (1987). Deterministic parsing and description theory. In P. Whitelock, M. M. Wood, H. L. Somers, R. Johnson, and P. Bennet (Eds.), *Linguistic Theory and Computer Applications*, pp. 69–112. London: Academic Press.

Matthews, C. (1993). Grammar frameworks in intelligent CALL. *CALICO Journal 11*(1), 5–28.

Menzel, W. and I. Schröder (1998a). Constraint-based diagnosis for intelligent language tutoring systems. In *IT&KNOWS Conference Proceedings: XV IFIP World Computer Congress*, Vienna and Budapest, pp. 484–497.

Menzel, W. and I. Schröder (1998b). Error diagnosis for language learning systems. In *NLP + AI 98 Conference Proceedings*, Volume 1, Moncton, Canada, pp. 45–51.

Miller, L. A., G. E. Heidorn, and K. Jensen (1981). Text-critiquing with the EPISTLE system: an author's aid to better syntax. In *AFIPS Conference Proceedings*, Arlington, VA, pp. 649–655. AFIPS Press.

Mogilevski, E. (1998). Le Correcteur 101. *CALICO Journal 16*(2), 183–196.

Mössenböck, H. (1993). *Object-Oriented Programming in Oberon-2.* Berlin: Springer-Verlag.

Mulford, G. W. (1989). Semantic processing for communicative exercises in foreign-language learning. *Computers and the Humanities 23*, 31–44.

Murphy, M., A. Krüger, and A. Grieszl (1998). RECALL — providing an individualized CALL environement. In Jager, Nerbonne, and van Essen (1998), pp. 203–222.

Nagata, N. (2002). BANZAI: An application of natural language processing to web-based language learning. *CALICO 19*(3), 583–599.

Nerbonne, J. (2003). Natural language processing in computer-assisted language learning. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*, Chapter 37, pp. 670–698. Oxford: Oxford University Press.

Nerbonne, J., D. Dokter, and P. Smit (1998). Morphological processing and computer-assisted language learning. *Computer Assisted Language Learning 11*(5), 543–559.

Nikitin, E. (1997). *Into the Realm of Oberon: An Introduction to Programming and the Oberon-2 Programming Language.* New York: Springer.

Oflazer, K. (1996). Error-tolerant finite-state recognition with apllications to morphological analysis and spelling correction. *Computational Linguistics 22*(1), 73–89.

Paramskas, D. (1993). ELAO: genèse et avenir. In P. Liddell (Ed.), *CALL: Theory and Application*, pp. 65–75. Victoria University Press.

Peterson, J. L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the ACM 23*(12), 676–687.

Pollard, C. J. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar.* Chicago: Chicago University Press.

Pollock, J. J. and A. Zamora (1984). Automatic spelling correction in scientific and scholarly text. *Communications of the ACM 27*(4), 358–368.

Pusack, J. P. (1984). Answer processing and error correction in foreign language CAI. In D. H. Wyatt (Ed.), *Computer-Assisted Language Instruction*, pp. 53–64. Oxford: Pergamon Press Ltd.

Reiser, M. and N. Wirth (1992). *Programming in Oberon: Steps beyond Pascal and Modula.* New York: ACM Press.

Richardson, S. and L. Braden-Harder (1993). The experience of developing a large-scale natural language processing system: Critique. In Jensen, Heidorn, and Richardson (1993), Chapter 7, pp. 77–89.

Richmond, I. M. (1999). Is your CALL connected? dedicated software vs. integrated CALL. In Cameron (1999), pp. 295–314.

Rypa, M. and K. Feurman (1995). CALLE: An exploratory environment for foreign language learning. In Holland, Kapland, and Sams (1995), Chapter 4, pp. 55–76.

Sams, M. R. (1995). Advanced technologies for language learning: The BRIDGE project within the ARI language tutor program. In Holland, Kapland, and Sams (1995), pp. 7–21.

Sanders, A. and R. Sanders (1987). Designing and implmenting a syntactic parser. *CALICO Journal 5*(1), 77–86.

Sanders, A. F. and R. H. Sanders (1989). Syntactic parsing: A survey. *Computers and the Humanities 23*, 13–30.

Sanders, R. (1991). Error analysis in purely syntactic parsing of free input: The example of German. *CALICO Journal 9*(1), 72–89.

Sanz, D. (1992). Grammaire: Quatre ténors à l'épreuve. *Science & Vie Micro 90*, 100–108.

Schneider, D. and K. F. McCoy (1998). Recognizing syntactic errors in the writing of secong language learners. In *Proceedings of Coling-ACL'98*, Volume 2, Montreal, Canada, pp. 1198–1204.

Schulze, M. (1998). Checking grammar — teaching grammar. *Computer Assisted Language Learning 11*(2), 215–227.

Schulze, M. and M.-J. Hamel (2000). Towards authentic tasks and experiences: The example of paser-based CALL. *The Canadian Journal of Applied Linguistics 3*(1-2), 79–90.

Schuster, E. (1986). The role of native grammars in correcting errors in second language learning. *Computer Intelligence 2*, 93–98.

Schwind, C. B. (1986). Overview of an intelligent language tutoring system. In *Proceedings of the 2nd International Conference on Artificial Intelligence CIIAM'86*, Paris, pp. 389–407. Hemès.

Schwind, C. B. (1988). Sensitive parsing: Error analysis and explanation in an intelligent language tutoring system. In *Proceedings of COLING 88*, Budapest, pp. 608–613.

Schwind, C. B. (1995). Error analysis and explanation in knowledge based language tutoring. *Computer Assisted Language Learning 8*(4), 295–324.

Sedgewick, R. (1988). *Algorithms* (2nd ed.). Reading, MA: Addison-Wesley.

Selva, T. and T. Chanier (2000). Génération automatique d'activités lexicales dans le système ALEXIA. *Sciences et Techniques Educatives 7*(2), 385–412.

Simard, M. and A. Deslauriers (2001). Real-time automatic insertion of accents in french text. *Natural Language Engineering 7*(2), 143–165.

Sleator and Temperley (1991). Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Pittsburg, PA.

Swartz, M. L. and M. Yazdani (Eds.) (1992). *Intelligent Tutoring Systems for Foreign Language Learning.* NATO ASI series F 80. Berlin: Spinger-Verlag.

Teixeira Martins, R., R. Hasegawa, M. D. G. Volpe Nunes, G. Montilha, and O. Novais De Oliveira, Jr. (1998). Linguistics issues in the development of ReGra: A grammar checker for Brazilian Portuguese. *Natural Language Engineering 4*(4), 287–307.

Tennant, H. (1981). *Natural Language Processing: An introduction to an emerging technology.* New York: PBI-Petrocelli Books, Inc.

Thurmair, G. (1990). Parsing for grammar and style checking. In *Proceedings of COLING-90*, Helsinki, pp. 365–370. ACL.

Tschichold, C. (1999). Grammar checking for CALL: Strategies for improving foreign language grammar checkers. In Cameron (1999), pp. 203–222.

Tschichold, C., F. Bodmer, E. Cornu, F. Grosjean, L. Grosjean, N. Kübler, and C. Tschumi (1994). Detecting and correcting errors in second language texts. *Computer Assisted Language Learning 7*(2), 151–160.

Tschumi, C. (1994). Les erreurs d'utilisation des temps anglais par les francophones: ébauche d'un vérificateur Prolog. *TRANEL 21*, 205–222.

Tschumi, C., F. Bodmer, E. Cornu, F. Grosjean, L. Grosjean, N. Kübler, and C. Tschichold (1994). The ARCTA prototype: An English writing tool and grammar checker for French-speakers. *TRANEL 21*, 223–228.

Uszkoreit, H. (1991). Strategies for adding control information to declarative grammars. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, pp. 237–245. ACL.

Vandeventer, A. (2001). Creating a grammar checker for CALL by constraint relaxation: A feasibility study. *ReCALL 13*(1), 110–120.

Véronis, J. (1988). Morphosyntactic correction in natural language interfaces. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Volume 2, pp. 708–713. ACL.

Vosse, T. (1992). Detecting and correcting morpho-syntactic errors in real texts. In *Third Conferece on Applied natural Language Processing: Proceedings*, Trento, Italy, pp. 111–118. ACL.

Wehrli, E. (1997). *Traitement automatique de la langue: Problèmes et méthodes.* Paris: Mason.

Weinberg, A., J. Garman, J. Martin, and P. Merlo (1995). A principled-based parser for foreign language tutoring in German and Arabic. In Holland, Kapland, and Sams (1995), pp. 23–44.

Weischedel, R. M. and J. E. Black (1980). Responding intelligently to unparsable inputs. *American Journal of Computational Linguistics 6*(2), 97–109.

Weischedel, R. M. and L. A. Ramshaw (1987). Reflections on the knowledge needed to process ill-formed language. In S. Nirenburg (Ed.), *Machine translation: Theoretical and methodological issues*, Studies in natural Language Processing, Chapter 10, pp. 155–167. Cambridge: Cambridge University Press.

Weischedel, R. M. and N. K. Sondheimer (1983). Meta-rules as a basis for procession ill-formed input. *American Journal of Computational Linguistics 9*(3–4), 161–177.

Weischedel, R. M., W. M. Voge, and M. James (1978). An artificial intelligence approach to language instruction. *Artificial Intelligence 10*, 225–240.

Winograd, T. (1983). *Language as a Cognitive Process*, Volume 1: Syntax. Reading, MA: Addison-Wesley.

Yang, J. C. and K. Akahori (1999). An evaluation of Japanese CALL systems on the WWW: Comparing a freely input approach with multiple selection. *Computer Assisted Language Learning 12*(1), 59–79.

Yazdani, M. (1991). The LINGER project: An artificial intelligence approach to second-language tutoring. *Computer Assisted Language Learning 4*(2), 107–116.

Yazdani, M. and J. Uren (1988). Generalising language-tutoring systems: A French/Spanish case study, using LINGER. *Instructional Science 17*, 179–188.