Design and Development of a System for the Detection of Agreement Errors in Basque

Arantza Díaz de Ilarraza, Koldo Gojenola and Maite Oronoz

Department of Computer Languages and Systems
University of the Basque Country
P.O. box 649, E-20080 Donostia
{jipdisaa, jibgogak, jiporanm}@si.ehu.es

Abstract. This paper presents the design and development of a system for the detection and correction of syntactic errors in free texts. The system is composed of three main modules: a) a robust syntactic analyser, b) a compiler that will translate error processing rules, and c) a module that coordinates the results of the analyser, applying different combinations of the already compiled error rules. The use of the syntactic analyser (a) and the rule processor (b) is independent and not necessarily sequential. The specification language used for the description of the error detection/correction rules is abstract, general, declarative, and based on linguistic information.

1 Introduction

The problem of the detection and correction of syntactic errors has been addressed since the early years of natural language processing. Different techniques (Vandeventer, 2003) have been proposed for the treatment of the significant portion of errors (typographic, phonetic, cognitive and grammatical) that result in valid words (Kukich, 1992). Although many commercial grammar checkers have been developed (Paggio and Music, 1998), there is little published work on their implementation or evaluation. This is due in part to the fact that the mechanisms used for the implementation have not been very sophisticated (as in some systems that use a large set of regular expressions) and also that commercial companies are not willing to reveal implementation details about their tools. The aim of the present work is to examine the feasibility of corpus-based syntactic error detection focusing in detecting agreement errors.

The system will be applied to Basque, an agglutinative language with relative free order among sentence elements. In our research group, work in error detection at morphological level has already been accomplished and a spelling checker-corrector (Aldezabal *et al.* 1999), - called XUXEN - was marketed 10 years ago. Error detection at syntactic level needs of a robust syntactic analyser and we will use the Basque syntactic analyser (Aduriz and Díaz de Ilarraza, 2003) that was developed using Constraint Grammar (CG) (Karlsson *et al.*, 1995).

Figure 1 shows the syntactic analysis chain in which sequential rule layers, most of them materialised in Constraint Grammars, enrich the output of the previous layer with the respective information.

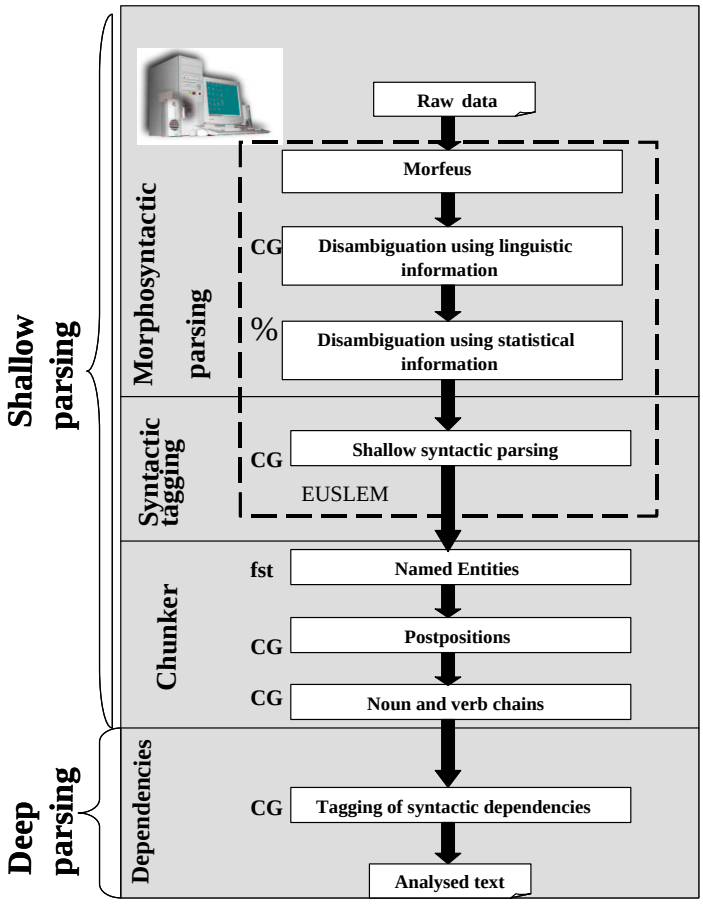

Fig. 1. The multi-layered syntactic analyser for Basque.

The parsing process starts with the outcome of the morphosyntactic analyser (Morfeus), which was created following the two-level morphology (Koskeniemi, 1983) and deals with all the lexical units of a text, both simple words and multiword units. The tagger/lemmatiser EUSLEM, not only obtains the lemma and category of each form but also includes a module for disambiguation. The disambiguation process is carried out by means of linguistic rules (CG) and stochastic rules based on Markovian models (Ezeiza, 2003). Once we have the morphosyntactic information by means of EUSLEM, the recognition of named entities and post-positional phrases is carried out. The subsequent level of chunking identifies verb and noun chains. The last step in this analysis chain is the identification of dependency relations among the components of the sentence in order to obtain a dependency syntactic tree. As a result of using the CG formalism, a lim-

ited amount of ambiguity remains, which reduces the number of parsing errors. However, this ambiguity is much lower than in other approaches such as CFG-based parsing systems, in which it is usual to encounter hundreds of parses for each sentence that are later discriminated using statistical information (Briscoe and Carroll, 2004). All the information in the analysis chain is interchanged by means of stardarised XML files (Artola *et al.*, 2004) and a class library for the management of all the linguistic information.

Not all the information provided by this analysis chain is necessary for the detection of some syntactic errors. For example, we can detect errors in ill composed postpositional phrases using pattern rules defined in CG and information only at morphosyntactic (word) level. Other kinds of errors such as agreement errors need all the available information, and specially the information related to the morphosyntactic analysis of each lexical unit and to the dependencies among the elements of the sentence.

The remainder of this paper is organised as follows. Section 2 describes the general architecture of the error processing system. In section 3 we will analyse the agreement errors found in a set of incorrect sentences. Section 4 describes the state of development of the application and we mention the main problems we will have to tackle. Finally some conclusions are outlined in section 5.

2 Architecture of the system

We have divided our system for agreement error detection in 4 independent modules (see figure 2). Each of the modules is explained in the following paragraphs.

2.1 Syntactic analysis

The syntactic analysis module that we already presented in figure 1 produces, for each input sentence, one or more dependency trees (see figure 3). Each dependency tree contains in each node information about the morphosyntactic analysis of each lexical unit, as well as the dependency relation with its parent and child nodes (e.g. subject, noun modifier, ...). Figure 3 shows the dependency tree of an incorrect sentence we took as an example. The subject *zentral nuklearrak* (nuclear power station), in the absolutive case, and the auxiliary verb, *dute*, which needs the subject to be in ergative case, do not agree. In the next section we will explain how we can represent this agreement error by means of a rule.

2.2 Error-rule compiler

We have defined a general specification language to be used to search for any linguistic structure, correct or not, in a dependency tree and to transform the obtained structure into a different one. Although this language allows to search for/transform sentences that fulfil a specific requirement, in this application we have used it to write error detection/correction rules.

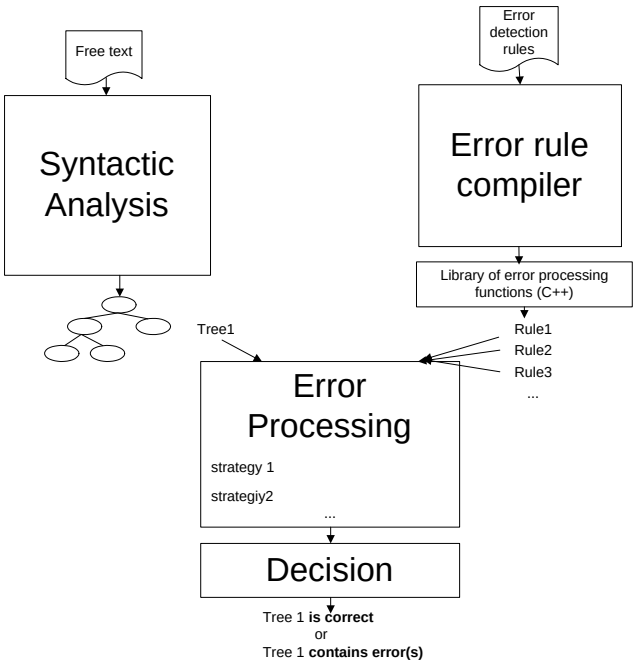

Fig. 2. Architecture of the system.

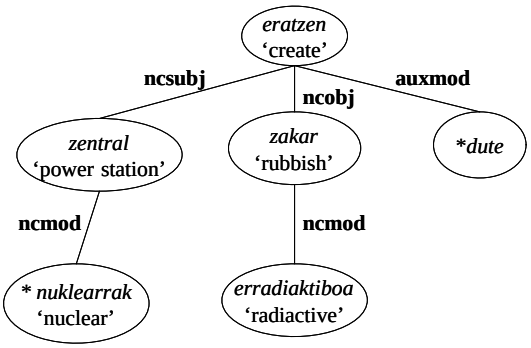

Fig. 3. Dependency tree for the sentence **Zentral nuklearrak zakar erradiaktiboa eratzen dute* (*Nuclear power station create radioactive rubbish).

The use of an abstract specification language has several advantages: a) declarativeness, b) maintainability and, c) efficiency, as the abstract rules will be compiled to an object language (C++).

Each rule contains four different sections (see figure 4): i) *Detect*: detection of the error in the dependency tree, ii) *Correct*: one or more possible ways to correct the error, iii) *Mark*: branches in the tree to be marked to represent the error and, iv) *Info*: message explaining the error.

This type of rule is very similar to the ones described in related systems (Knutsson *et al.*, 2001). In the following paragraphs we will describe each component in detail.

```
RULE AGREEMENT_SUBJ_CASE_NOR_NORK
(
      Detect (
            @!ncsubj!ncmod∼ &
            @!auxmod.type == 'nor-nork' &
            @!ncsubj!ncmod.case != @!auxmod.nork.case
            )
      Correct (
            (@!auxmod.nork.case := @!ncsubj!ncmod.case)
            # Zentral nuklearrAK zakar erradiaktiboa eratzen DU.
            # 'The nuclear power station creates radioactive rubbish'
            |
            (@!ncsubj!ncmod.case := @!auxmod.nork.case
            # Zentral nuklearrEK zakar erradiaktiboa eratzen DUTE.
            # 'Nuclear power stations create radioactive rubbish'
            )
      Mark ((ncsubj & auxmod))
      Info (The subject and the auxiliary verb do not agree in case.)
            )
```

Fig. 4. Example of a rule.

Error detection The error processing rules allow the traversing of the dependency tree while at the same time checking syntactic constraints.

In the description of the errors we use linguistic information such as tags that define dependency relations between the elements of the sentence (e.g. ncsubj, ncobj,...), as well as tags defining features of the syntactic elements (number, case, ...). Apart from this, some operators have been defined, i) to describe the traversal across the branches of the dependency tree (e.g. '@' indicates the current tree node the program is inspecting, '!' followed by a dependency tag (e.g. ncsubj) crosses a dependency link down the current node, '¡' ascends a

dependency link, ...) and, ii) to inspect linguistic features (e.g. '∼' looks for the existence of a feature, '#' asks for the number of dependency tags,...).

Thus, for example, the *Detect* part of the example in figure 4 can be paraphrased as: starting from the current node ('@'), descend in the tree ('!') across the branch tagged with the *'ncsubj'* dependency relation and descend again ('!') across the branch tagged with *'ncmod'*. An agreement error between the subject and the verb occurs if the case of the subject and the 'case' of the agreement marker in the auxiliary verb are different (*'!='*).

Error correction The *Correct* part of the rule contains two possible corrections: it can be corrected by assigning the case of the subject to the auxiliary verb so that we obtain the acceptable sentence *'Zentral nuklearrAK zakar erradiaktiboa eratzen DU'* (THE nuclear power station createS radioactive rubbish). Another option ('|') could be to assign the 'case' of the auxiliary verb to the subject and obtain the correct sentence *'Zentral nuclearrEK zakar erradiaktiboa eratzen DUTE'* (Nuclear power stationS create radioactive rubbish).

Morphological generation will be used to obtain these two correct sentences. One of them will fit in better than the other one in the text. For the process of discriminating among candidate corrections, several methods have been proposed. Some of them are based on heuristics regarding the number of changes required at the morphosyntactic level (Menzel, 1988) or at the semantic and phonetic levels (Genthial *et al.*, 1994). Some other methods take into account the syntactic/semantic context of the incorrect element (Golding and Roth, 1996, Carlsson *et al.* 2001). In this last paper *'context sensitive text correction'* is used when dealing with spelling errors in order to choose the best candidate. We propose a similar technique but at a syntactic level to choose the best option among all the candidate corrections. It is important to remark that after applying morphological generation to create all the possible corrections, a reparsing of the resulting dependency trees would be necessary to test whether the introduction of a correction does not produce any other errors.

At the moment, a number of rules have been designed, and only the section relative to *detection* has been completely implemented. As the rules are written in an abstract language, they cannot be directly applied to a dependency tree because they must first be translated into executable statements. We defined and implemented a syntax-directed translation scheme (Aho *et al.*, 1985) for that purpose. A lexical analyser recognises lexical units in the rules and a syntactic analyser analyses the correct syntax of the rule and generates the code in C++. Once the code is created, the error processing module will apply the executable rules to the trees.

2.3 Error processing module

The next step in the process of error detection is the application of the rules to the trees. Some strategies will be defined for that purpose. The simplest strategy could be to 'apply all the rules to all the nodes in the tree'. We have

manually analysed 64 sentences containing agreement errors and have noticed that the structures in which the error detection rules are applied are repeated. This phenomenon could be reflected in a strategy by applying first the rules that match up these structures. In a near future, different strategies will be defined and evaluated taking into consideration aspects such as the minimisation of morphological and structural changes, or the introduction of new errors.

2.4 Decision module

The problem when dealing with syntactic ambiguity lies in deciding which of the different analyses is the correct one. In case of ambiguity, the error detection system will have more than one dependency tree for a sentence. When the error detection rules are applied to them, we should study what happens when an error is detected. It may happen that, in an ambiguous correct sentence, the error detection rules could mark one of the incorrect analyses as a syntactic error, giving a false alarm. In these cases, we think that an error should only be marked when all the analyses contain an error. The decision module will be in charge of this task.

3 Analysis of agreement errors

In order to define the error rules to be used for agreement error detection, first we manually analysed a set of 64 sentences with some type of agreement error. These sentences were manually extracted from texts of students that have Basque as their mother language. Most of the sentences are related to scientific issues (e.g. computer science students' final year projects,...).

In that corpus, we found 66 errors in 64 sentences (2 sentences with 2 errors). We want to notice that other 2 sentences have been rejected since the agreement error was difficult to detect. One of them is a relative clause with ellipsis that we want to analyse in depth. The second one is a sentence with a non-finite verb, which do not carry any mark to indicate tense or person. As the number of non-finite verbs in this corpus is small, we have decided to start with the finite verbs and try using verb subcategorisation for non-finite verbs later.

Regarding agreement errors, we have made a distinction taking into account the linguistic context in which they occur. Thus, we differentiate between:

- Agreement inside noun or verb chains (5 errors from 64, 7.8 %).
- Agreement inside clauses (59 errors from 64, 92.2 %).
- Agreement between subordinate and main clauses in the sentence (0 errors).

We can see that the number of agreement errors inside clauses is high, so we have focused our analysis in that kind of error.

With the aim of better understanding the high number of errors into clauses, we will briefly explain this kind of agreement. In finite verbs, the agreement elements are marked explicitly in the following way: the verb agrees with the

subject, object or indirect object of the sentence. These elements can appear in any order in the sentence, and each of them must agree with their corresponding agreement markers in the verb morphemes in number and person. This is a source of many syntactic errors, considerably higher than in languages with a more reduced kind of agreement, as English or Spanish. Basque is a morphologically ergative language with accusative syntax. Morphological ergativity implies that the subject of a transitive verb is realized in the ergative case, as opposed to the object of a transitive verb, and the subject of an intransitive verb, which are realized in the absolutive case. Besides, the indirect object is realized in the dative case. Some of the rules we have defined for agreement error detection are based on these characteristics. They look for disagreement in case or number between the agreement markers of the auxiliary verbs and finite lexical verbs and the subject, object and indirect object.

Table 1 indicates the number of errors found in each of the mentioned categories.

	Elements of the sentence		
Verb	Subject	Object	Indirect object
In agreement markers in 'case'	30	0	1
In agreement markers in number	17	4	3
Total	55		

Table 1. Number of errors in each category.

As we can see in the table above, 30 errors (54.6% of the total) are due to disagreement between the verb and the subject. The reason for this phenomena is that in Basque the morpheme for the absolutive plural and the one for the ergative singular are identically written.

The remaining 4 errors are divided as follows: 3 are due to the fact that if the object of the sentence is declined in the partitive case, the correspondent agreement mark in the verb must be singular. In the other error, an object appears in the sentence with an intransitive verb.

28 rules have been already designed for detecting these errors. Most of them check general structures and detect a high number of errors (e.g. 24 errors are detected with 3 rules) while others are for very specific structures.

At the moment, we are analysing different possibilities of dealing with several phenomena such as some types of ellipsis, relative clauses and coordination.

4 The state of development

At this point, the state of the work we present in the paper is the following:

1. The syntactic parser is almost complete.

2. The error detection rule compiler is finished and running while the error correction compiler is still in the design phase.
3. The rule application module is operational.
4. 64 sentences with agreement errors have been analysed, and the corresponding rules designed.
5. The first experiments have been manually developed with good results.

Regarding evaluation, we have in mind a type of assessment process similar to that of Starlander *et al.* (2002). It is to be expected that some of the errors in the evaluation will be due to the problems described in the following paragraphs.

When developing a system to cope with syntactic errors in real texts, we think that one of the main problems we will have to deal with is the lack of coverage of the syntactic analyser.

Since the parsing system we are using contains several modules, each of them could add a limited amount of errors, which would be accumulated through each phase. As each module can introduce new errors, this could increase the number of false alarms, where a correct sentence is marked as incorrect due to errors in any of the previous phases. In fact, it could be paradoxical to have the fact that the analysed sentences contain more errors due to correct sentences incorrectly analysed than to real syntactic errors. This fact makes compulsory a corpus-based evaluation (Gojenola and Oronoz, 2000), in order to obtain high precision and minimise the number of false alarms. We also expect that this will be done at the cost of lowering recall, only marking errors that can be detected with very high precision.

5 Conclusions

We have presented a system for the treatment of complex syntactic errors based on three main modules: a robust syntactic analyser, an error processing rule compiler and a coordination module that will give way to experiment with different strategies to error detection and correction. The goal is to process real texts with high precision error detection/correction, minimising false alarms, which are the main bottleneck in current grammar checking systems. The system will be mainly declarative, as all the modules are based on abstract views of the syntactic processes involved (syntactic analysis and error rules), and extensible, being based on an object oriented approach. At the moment, the syntactic analysis module and the error rule application module are operational. Regarding the error rule compiler, the error detection part is already implemented, and a number of rules have been devised and tested on real sentences.

6 Aknowledgments

This research is supported by the University of the Basque Country(9UPV00141.226-146012002) and the Ministry of Industry of the Basque Government (XUXENG project, 0D02UN52). Thanks to Ruben Urizar for his help writing the final version of the paper.

References

Aduriz, I., Díaz de Ilarraza, A.: Morphosyntactic Disambiguation and Shallow Parsing in Computational Processing of Basque. Inquiries into the Lexicon-syntax Relations in Basque (2003) 1–21 Bernard Oyharçabal Ed. University of the Basque Country, Bilbao.

Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Reading, Mass.: Addison-Wesley (1985)

Aldezabal I., Alegria I., Ansa O., Arriola J., Ezeiza N.: Designing Spelling Correctors for Inflected Languages Using Lexical Transducers In: Proceedings of EACL'99, 265-266. Bergen, Norway. 8-12 June 1999.

Artola X., Díaz de Ilarraza A., Ezeiza N., Gojenola K., Sologaistoa A., Soroa, A.: Eulia: a Graphical Web Interface for Creating, Browsing and Editing Linguistically Annotated Corpora. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation, Lisbon, Portugal (2004)

Briscoe, E., Carroll, J.: Robust Accurate Statistical Annotation of General Text. In: Proceedings of the 3rd International Conference on Language Resources and Evaluation, Las Palmas, Gran Canaria (2002) 1499–1504

Carlson, A.J., Rosen, J., Roth, D.: Scaling Up Context-sensitive Text Correction. In: Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference, AAAI Press (2001) 45–50

Ezeiza, N.: Corpusak Ustiatzeko Tresna Linguistikoak. Euskararen Etiketatzaile Sintaktiko Sendo eta Malgua. PhD thesis. University of the Basque Country, Donostia (2003)

Genthial, D., Courtin, J., Menezo J.: A More User-friendly Correction. In: COLING-94, Tokyo. (1994)

Gojenola, K., Oronoz, M.: Corpus-based Syntactic Error Detection Using Syntactic Patterns. In: NAACL-ANLP00,Student Research Workshop. (2000)

Golding, A.R., Roth, D.: Applying Winnow to Context-sensitive Spelling Correction. In: Proc. 13th International Conference on Machine Learning, Morgan Kaufmann (1996) 182–190

Karlsson, F., Voutilainen, A., Heikkila, J., Anttila, A.: Constraint Grammar: Language-independent System for Parsing Unrestricted Text. Prentice-Hall, Berlin (1995)

Knutsson, O., Carlberger, J., Kann, V.: An Object-oriented Rule Language for High-level Text Processing. In: Poster, NoDaLiDa '01 - 13th Nordic Conference on Computational Linguistics. (2001)

Koskenniemi, K.: Two-level Morphology: a General Computational Model for Word-form Recognition and Production. University of Helsinki, Helsinki (1983)

Kukich, K.: Techniques for Automatically Correcting Words in Text. ACM Computing Surveys **24** (1992) 377–439

Menzel, W.: Error Diagnosing and Selection in a Training System for Second Language Learning. COLING-88 (1988) 414–419

Paggio, P., Music, B.: Evaluation in the Scarrie Project. In: Proceedings of the First International Conference on Language Resources & Evaluation, Granada, Spain (1998) 277–282

Starlander, M., Popescu-Belis, A.: Corpus-based Evaluation of a French Spelling and Grammar Checker. In: Proceedings of the Third International Conference on Language Resources and Evaluation. (2002) 268–274

Vandeventer, A.: Syntactic Error Diagnosis in the Context of Computer Assisted Language Learning. PhD thesis. Universite de Geneve, Geneve (2003)