Corpus-Based Syntactic Error Detection Using Syntactic Patterns

Koldo Gojenola, Maite Oronoz
Informatika Fakultatea, 649 P. K., Euskal Herriko Unibertsitatea,
20080 Donostia (Euskal Herria)
jipgogak@si.ehu.es, jiboranm@si.ehu.es

Abstract

This paper presents a parsing system for the detection of syntactic errors. It combines a robust partial parser which obtains the main sentence components and a finite-state parser used for the description of syntactic error patterns. The system has been tested on a corpus of real texts, containing both correct and incorrect sentences, with promising results.

Introduction

The problem of syntactic error detection and correction has been addressed since the early years of natural language processing. Different techniques have been proposed for the treatment of the significant portion of errors (typographic, phonetic, cognitive and grammatical) that result in valid words (Weischedel and Sondheimer 1983; Heidorn *et al*. 1982). However, although most currently used word-processors actually provide a grammar checking module, little work has been done on the evaluation of results. There are several reasons for this:

- Incomplete coverage. Some of the best parsers at the moment can analyze only a subset of the sentences in real texts. Compared to syntactic valid structures, the set of syntactically incorrect sentences can be considered almost infinite. When a sentence cannot be parsed it is difficult to determine whether it corresponds to a syntactic error or to an uncovered syntactic construction. In the literature, syntactic errors have been defined mostly with respect to their corresponding correct constructions. The use of unrestricted corpora confronts us with the problem of flagging a correct structure as erroneous (false alarms). These facts widen the scope of the problem, as not only incorrect structures but also correct ones must be taken into account.

  On the other hand, robust parsing systems (e.g., statistical ones) are often unable to distinguish ungrammatical structures from correct ones.
- The need for big corpora. Each kind of syntactic error occurs with very low frequency and, therefore, big corpora are needed for testing. Even if such corpora were available, the task of recognizing error instances for evaluation is a hard task, as there are no syntactically annotated treebanks with error marks for the purposes of evaluation and testing. Thus, to obtain naturally occurring test data, hundreds of texts must be automatically and manually examined and marked.

The aim of the present work is to examine the feasibility of corpus-based syntactic error detection, with methods that are sensitive enough to obtain high correction rates and discriminating enough to maintain low false alarm rates. The system will be applied to Basque, an agglutinative language with relative free order among sentence components. Its recent standardization makes it necessary to develop a syntactic checking tool.

The remainder of this paper is organized as follows. After commenting on the literature on syntactic error detection in section 2, section 3 presents a description of the linguistic resources we have used. Section 4 describes the error types we have treated, while section 5 gives the evaluation results.

1 Background

Kukich (1992) surveys the state of the art in syntactic error detection. She estimates that a proportion of all the errors varying between 25% and over 50%, depending on the application, are valid words. Atwell and Elliott (1987) made a manual study concluding that 55% of them are local syntactic errors (detectable by an examination of the local syntactic context), 18% are due to global syntactic errors (involving long-distance syntactic dependencies, which need a full parse of the sentence), and 27% are semantic errors. Regarding their treatment, different approaches have been proposed:

- The relaxation of syntactic constraints (Douglas and Dale 1992). This grammar-based method allows the analysis of sentences

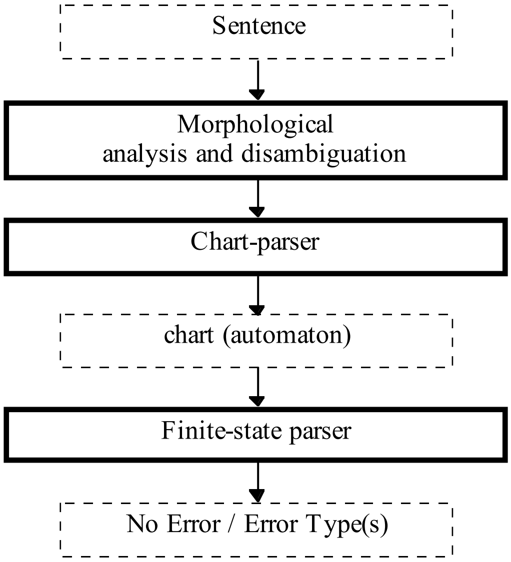

Figure 1. Overview of the system.

that do not fulfill some of the constraints of the language by identifying a rule that might have been violated, determining whether its relaxation might lead to a successful parse. Its main disadvantage is the need of a full-coverage grammar, a problem not solved at the moment, except for restricted environments (Menzel and Schröder 1999).

- Error patterns (Kukich 1992; Golding and Schabes 1996; Mangu and Brill 1997), in the form of statistical information, hand-coded rules or automatically learned ones.
- Charts have been used in grammar-based systems as a source of information; they can be resorted to if no complete analysis is found, so as to detect a syntactic error (Mellish 1989; Min-Wilson 1998).

2 Linguistic resources

We have used a parsing system (Aldezabal *et al.* 1999, 2000) divided in three main modules (see figure 1):

- Morphological analysis and disambiguation. A robust morphological analyzer (Alegria *et al.* 1996) obtains for each word its segmentation(s) into component morphemes. After that, morphological disambiguation (Ezeiza *et al.* 1998) is applied, reducing the high word-level ambiguity from 2.65 to 1.19 interpretations.
- Unification-based chart-parsing. After morphological analysis and disambiguation, a PATR-II unification grammar is applied bottom-up to each sentence, giving a chart as a result. The grammar is partial but it gives a complete coverage of the main sentence elements, such as noun phrases, prepositional phrases, sentential complements and simple sentences. The result is a shallow parser (Abney 1997) that can be used for subsequent processing (see figure 2). In this figure, dashed lines are used to indicate lexical elements (lemmas and morphemes), while plain lines define syntactic constituents. Bold circles represent word-boundaries, and plain ones delimit morpheme-boundaries. The figure has been simplified, as each arc is actually represented by its morphological and syntactic information, in the form of a sequence of feature-value pairs.
- Finite-state parsing. A tool is needed that will allow the definition of complex linguistic error patterns over the chart. For that reason, we view the chart as an automaton to which finite-state constraints can be applied encoded in the form of automata and transducers (we use the Xerox Finite State Tool, XFST, (Karttunen *et al.* 1997)). Finite-state rules provide a modular, declarative and flexible workbench to deal with the resulting chart. Among the finite-state operators used, we apply composition, intersection and union of regular expressions and relations.

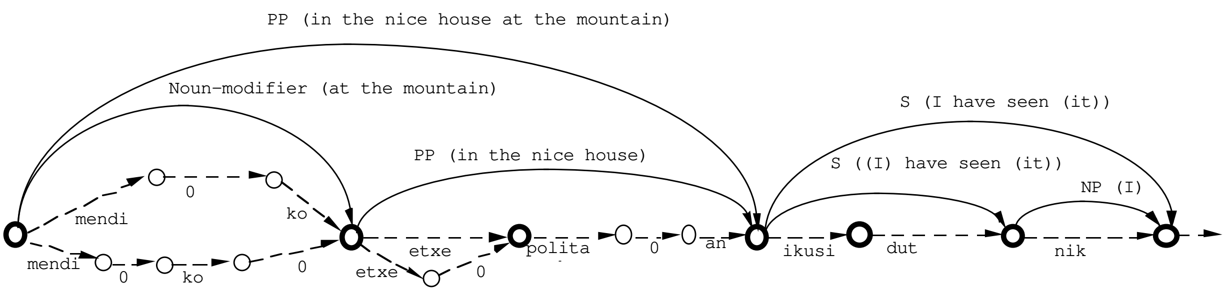

Figure 2. State of the chart after the analysis of *Mendiko etxe politean ikusi dut nik* ('I have seen (it) in the nice house at the mountain').

Durangon,	*1999ko*	*martxoaren*	*7an*
In Durango,	1999,	March	the 7th
(Durango, inessive, sing)	(1999 genitive)	(March, genitive, sing)	(7, inessive sing)

Example 1. Format of a valid date expression.

The full system provides a robust basis, necessary for any treatment based on corpora. In the case of error detection, a solid base is indispensable.

3 Error detection

As a test, we chose the case of date expressions due to several reasons:

- It was relatively easy to obtain test data compared to other kinds of errors. Although the data must be obtained mostly manually, date expressions contain several cues (month names, year numbers) that help in the process of finding semiautomatically test sentences. In any case, manual marking is needed for all the retrieved sentences.
- The context of application is wide, that is, date expressions contain morphologically and syntactically rich enough phenomena where several types of errors can be found. These can be viewed as representative of the set of local syntactic errors so that the same procedure can be used when dealing with other kinds of errors. Example 1 shows one of the formats of a date expression.

Basque being an agglutinative language, most of the elements appearing in date expressions (year numbers, months and days) must be inflected, attaching to them the corresponding number and case morphemes. Moreover, each different date format requires that the elements involved appear in fixed combinations. This is a common source of errors, not detectable by a spelling-checker, as each isolated word-form is correct.

For evaluation, we collected 267 essays written by students (with a high proportion of errors) and texts from newspapers and magazines, totaling more than 500,000 words. From them we chose 658 sentences, including correct dates, incorrect dates, and also structures 'similar' to dates (those sentences containing months and years, which could be mistaken for a date), in order to test false positives (see table 1). As a result of the selection procedure, the proportion of errors is higher than in normal texts. We divided our data into two groups. One of them was used for development, leaving the second one for the final test. The proportion of correct dates is higher in the case of test data with respect to those in the development corpus, so that the effect of false positives will be evaluated with more accuracy.

	Development corpus		Test corpus	
Number of sentences	411		247	
Correct dates	65		39	
Structures 'similar' to dates	255		171	
Incorrect dates	91		37	
Incorrect dates with 1 error	43	% 47	6	% 16
Incorrect dates with 2 errors	42	% 46	27	% 73
Incorrect dates with 3 errors	6	% 7	4	% 11

Table 1. Test data.

Error type	Example
1. The year number cannot be inflected using a hyphen	Donostian, 1995-eko martxoaren 14an
2. The month (*martxoak*) must appear in lowercase	1997ko martxoak 14
3. The optional locative preceding dates (*Frantzia*) must be followed by a comma	Frantzia 1997ko irailaren 8an
4. The day number after a month in genitive case (*martxoaren*) must have a case mark	Donostian, 19995eko martxoaren 22
5. The day number after a month in absolutive case (*ekainak*) cannot have a case mark	1998.eko ekainak 14ean argitaratua
6. The month (*martxoan*) must be inflected in genitive or absolutive case	Donostian, 1995.eko martxoan 28an
Combination of errors (2, 3 and 4)	karrera bukatu nuenean 1997ko Ekainaren 30an

Table 2. Most frequent error types in dates.

```
define NP_Month_Absolutive_or_Ergative   ...
define PP_Year_Genitive   ...
define Error_Type_5        NP_Month_Absolutive_or_Ergative   Inflected_Number;
define Mark_Error_Type_5   [Error_Type_5]   @->   BEGINERRORTYPE5   "..."   ENDERRORTYPE5
                           || Optional_Place_Name   Optional_Comma   PP_Year_Genitive   _  ;
```

Example 2. Regular expressions for an error pattern.

After examining different instances of errors, we chose the six most frequent error types (see table 2). In a first phase, one or more patterns were defined for each error type. However, we soon realized that this approach failed because quite often two or three errors might appear in the same expression. This phenomenon asked for a kind of 'gradual relaxation' approach, which had to consider that several mistakes could co-occur. Instead of treating each error independently, we had to design error patterns bearing in mind not only the correct expression, but its erroneous versions as well. For example, the last sentence in table 2 contains three different errors, so that the error pattern for the second error should consider the possibility of also containing errors 3 and 4. This relaxation on what could be considered a correct date had the risk of increasing the number of false positives. As the number of interactions among errors grows exponentially with the number of errors (there are potentially 2^6 combinations of the six error types), we based our error patterns on the combinations actually found in the corpus, so that in practice that number can be considerably reduced (we did not find any expression containing more than three errors in the corpus).

The error pattern for the fifth kind of error (see example 2[1]) is defined in two steps. First, the syntactic pattern of the error is defined (an NP consisting of a month in ergative or absolutive case followed by an inflected number), and named *Error_Type5*. Second, a transducer (*Mark_Error_Type_5*) is defined which surrounds the incorrect pattern (represented by "...") by two error tags (BEGINERRORTYPE5 and ENDERRORTYPE5). To further restrict the application of the rule, left and right contexts for the error can be defined (in a notation reminiscent of two-level morphology), mostly to assure that the rule is only applied to dates, thus preventing the possibility of obtaining false positives.

Concerning the definition of error patterns, equal care must be taken for correct and incorrect dates. In a first phase, we devised rules for the errors but, after testing them on correct dates from the development corpus, we had to extend the rules so as to eliminate false positives. As a result, more than 60 morphosyntactic patterns (each corresponding to a finite-state automata or transducer) were needed for the definition of the six basic error patterns. They range from small local constraints (45 automata with less than 100 states) to the most complex patterns (a transducer with 10,000 states and 475,000 arcs).

4 Evaluation

Table 3 shows the results. As the development corpus could be inspected during the refinement of the parser, the results in the second and third columns can be understood as an upper limit of the parser in its current state, with 100% precision (no false alarms) and 91% recall.

The system obtains 84% recall over the corpus of previously unseen 247 sentences. 31 errors out of 37 are detected giving the exact cause of the error (in cases with multiple errors almost all of them were found).

	Development corpus		**Test corpus**	
Number of sentences	411		247	
Undetected date errors	7	9%	6	16%
Detected date errors	84	91%	31	84%
False alarms	0		5	

Table 3. Evaluation results.

[1] For more information on XFST regular expressions, see (Karttunen *et al.* 1997).

Example	Cause of the error
atxiloketa 1998ko urtarriletik irailaren 16ra ... *the imprisonment from January 1998 till the 16th of September*	Structure similar to a date incorrectly interpreted as a date and flagged as erroneous.
Donostian 1960ko Urtarrilaren jaioa *born in Donostia in the January of 1960*	Incorrect Basque construction that is interpreted as a date.
etorriko da 1997ko irailaren 26ko 1:15etan *it will come the 26 of September 1997 at 1:15*	The system takes the hour number (1:15) as the day of the month.
atzotik 1999ko abenduaren 31 arte *from yesterday until the 31st of December*	The grammar does not cover the *arte* (until) particle, so a correct date is flagged as ungrammatical.
Primakovek 1998ko irailaren 11n hartu zuen ... *Primakov took it on the 11th of September 1998*	The unknown word *Primakov* is interpreted as a locative.

Table 4. False alarms.

Regarding precision, there are 5 false alarms, that is, correct dates or sentences similar to dates flagged as erroneous. If these false positives are divided by the number of sentences (247) of the test corpus, we can estimate the false alarm rate to be 2.02% over the number of dates in real texts. Table 4 examines some of the false alarms, two of them due to expressions similar to dates that are mistaken for dates, other two relate to constructions not taken into account in the design of the partial grammar, and the last one is due to insufficient lexical coverage.

Although the results are promising, more corpus data will be needed in order to maximize precision.

Conclusions

This work presents the application of a parsing system to syntactic error detection. The reported experiment has as its main features:

- It is corpus-based. If a system is to be useful, it must be tested on real examples of both correct and incorrect sentences. Although this may seem evident, it has not been the case for most of the previous work on syntactic errors. This implies the existence of big corpora and, for most of the errors, manual annotation.
- The most successful methods for error detection, i.e., relaxation of syntactic constraints and error patterns over a chart, have been combined with good results. On the other hand, the relaxation is not applied dynamically at parsing time, but it has been manually coded. This implies a considerable amount of work, as we had to consider the formats for valid sentences as well as for all their incorrect variants.
- A partial robust parsing architecture provides a powerful way to consider simultaneously information at the morphemic and syntactic levels. The unification grammar is necessary to treat aspects like complex agreement and word order variations, currently unsolvable using finite-state networks. It constructs all the possible syntactic components. On the other hand, regular expressions in the form of automata and transducers are suitable for the definition of complex error patterns based on linguistic units.

We are currently exploring new extensions to the system:

- Adding new kinds of errors. Our system, as well as any system dealing with syntactic errors, suffers the problem of scaling up, as the addition of new types of errors will suppose an increment in the number of error patterns that involves a considerable amount of work in the process of hand-coding the rules. The possible interaction among rules for different error types must be studied, although we expect that the rule sets will be mostly independent. Another interesting aspect is the reusability of the linguistic patterns: in the process of treating errors in dates some patterns describe general linguistic facts that can be reused, while others pertain to idiosyncratic facts of dates.

  We plan to extend the system to other qualitatively different types of errors, such as those involving agreement between the main components of the sentence, which is very rich in Basque, errors due to incorrect use of subcategorization and errors in post-positions. Although the number of potential syntactic errors is huge, we think that the treatment of the most frequent kinds of error with high recall and precision can result in useful grammar-checking tools.
- Automatic acquisition of error detecting patterns. Although manual examination seems unavoidable we think that, with a corpus of errors big enough, machine learning techniques could be applied to the

problem of writing error patterns (Golding and Roth 1996; Mangu and Brill 1997). This solution would be even more useful in the case of combinations of different errors. In any case, it must be examined whether automatic methods reach the high precision and reliability obtained by hand-coded rules.

- Using either hand-coded rules or automatically learned ones, both methods have still the problem of obtaining and marking big test corpora, a process that will have to be made mostly manually (except for some limited cases like word confusion (Golding and Roth 1996)). This is one of the major bottlenecks.

Acknowledgements

This research is supported by the Basque Government, the University of the Basque Country and the Interministerial Commission for Science and Technology (CICYT). Thanks to Gorka Elordieta for his help writing the final version of the paper.

References

Agirre E., Gojenola K., Sarasola K., Voutilainen A. (1998) *Towards a Single Proposal in Spelling Correction.* COLING-ACL'98, Montreal.

Abney S. (1997) *Part-of-Speech Tagging and Partial Parsing*. In Corpus-Based Methods in Language and Speech Processing, Kluwer, Dordrecht, 1997.

Aldezabal I., Gojenola K., Oronoz M. (1999) *Combining Chart-Parsing and Finite State Parsing*. Proceedings of the Student Session of the European Summer School in Logic, Language and Computation (ESSLLI'99), Utrecht.

Aldezabal I., Gojenola K., Sarasola K. (2000) *A Bootstrapping Approach to Parser* Development. Sixth International Workshop on Parsing Technologies, Trento.

Alegria I., Artola X., Sarasola K., Urkia. M. (1996) *Automatic morphological analysis of Basque*. Literary & Linguistic Computing, Vol. 11.

Atwell E., Elliott S. (1987) *Dealing with Ill-Formed English Text*. In The Computational Analysis of English: a Corpus-Based Approach, De. Longman.

Douglas, S., Dale R. 1992. *Towards Robust PATR*. COLING'92, Nantes.

Ezeiza N., Alegria I., Arriola J.M., Urizar R., Aduriz I. (1998) *Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages*. COLING-ACL-98, Montreal.

Golding A. and Schabes. Y. (1996) *Combining trigram-based and feature-based methods for context-sensitive spelling correction.* In Proc. of the 34th ACL Meeting, Santa Cruz, CA.

Golding A., Roth. D. (1996) *A Winnow-based Approach to Spelling Correction*. Proceedings of the 13th International Conference on Machine Learning, ICML'96.

Heidorn G. E., Jensen K., Miller L. A., Byrd R. J., Chodorow M. S. (1982) *The EPISTLE text-critiquing system*. IBM Systems Journal, Vol. 21, No. 3.

Karttunen L., Chanod J-P., Grefenstette G., Schiller A. (1997) *Regular Expressions For Language Engineering*. Journal of Natural Language Engineering.

Kukich K. (1992) *Techniques for automatically correcting words in text*. In ACM Computing Surveys, Vol. 24, N. 4, December, pp. 377-439.

Mangu L., Brill E. (1997) *Automatic Rule Acquisition for Spelling Correction*. Proceedings of the 14th International Conference on Machine Learning, ICML'97.

Mellish C. (1989) *Some Chart-Based Techniques for Parsing Ill-Formed Input*. EACL'89.

Menzel W., Schröder I. (1999) *Error Diagnosis for Language Learning Systems*. ReCALL, special edition, May 1999.

Min K., Wilson W. (1998) *Integrated Control of Chart Items for Error Repair*. COLING-ACL'98, Montreal.

Roche E., Schabes Y. (1997) *Finite-State Language Processing*. MIT Press.

Weischedel R.M., Sondheimer N.K. (1983) *Meta-rules as a Basis for Processing Ill-Formed Input*. American Journal of Computational Linguistics, 9.