NLP-enhanced error checking for Catalan unrestricted text

Toni Badia, Àngel Gil, Martí Quixal, Oriol Valentín

Grup de Lingüística Computacional
Universitat Pompeu Fabra
Rambla, 30-32
E-08003 Barcelona
{toni.badia,angel.gils,marti.quixal,oriol.valentin}@upf.edu

Abstract

We present here a general-purpose spell and grammar error detection architecture for Catalan unrestricted text. This architecture is based on a previous existing shallow morphosyntactic parser, which had to be adapted in order to successfully handle ill-formed input. The goal of this research is to obtain an architecture that can be used for developing morphosyntactic error checkers for both native and non-native speakers. We briefly present how we are currently customizing such an architecture in two different projects, as well as a means for annotating and exploiting error corpora (which ultimately condition the implementation of error checkers). We conclude with some remarks and future work.

1. Introduction

We present here how we work on the adaptation of a general-purpose natural language processing architecture in order to make it useful for the task of error detection in a broad sense. Lower level errors (orthographical and morphosyntactic ones) are handled with almost no restrictions, whereas errors requiring more complex analyses must be handled with domain-specific modules.

The architecture we departed from was CATCG, a modular general-purpose shallow morphosyntactic parser for unrestricted Catalan text (Badia et al., 2001). The components of this architecture are a preprocessing module, a morphologic analyser and a shallow syntactic analyser, as shown in Figure 1.

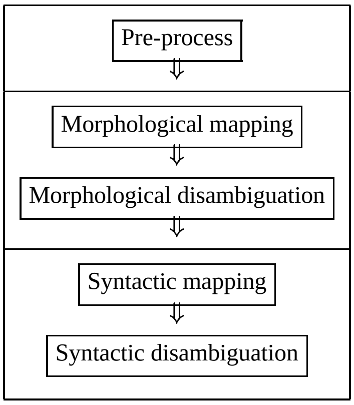

Figure 1: CATCG –previous existing architecture

The first two components of Figure 1 are implemented in Perl and fed with a set of word-form dictionaries; morphological tag mapping is performed independently of the context. The remaining three components of Figure 1 are implemented using the Constraint Grammar (CG) formalism –(Karlsson et al., 1995) and (Tapanainen, 1996). The CG formalism is a constraint-based (basically complex regular expressions) system that operates in a two-way step. First, a step called mapping, each word in a text is assigned all its possible interpretations (basically word class and related morphological information in the morphological mapping module, and syntactic function in the syntactic mapping module). Second, the disambiguation step, which consists in deleting inadequate interpretations with the proviso that no real ambiguities are discarded. In our system morphological mapping is performed outside the CG formalism. As for the disambiguation, linguistics-based rules remove or select the corresponding readings given a context –see details in (Badia et al., 2001).

2. An architecture for morphosyntactic error detection

Generally speaking the adaptation of the architecture in Figure 1 consists in (i) introducing an orthographic error detection module right after the morphological mapping, (ii) the relaxation of the morphosyntactic disambiguation module, and (iii) the incorporation of several error detection modules interspersed among the modules of the previous architecture. As a result, we obtained an architecture as the one reflected in Figure 2. New modules are written in small capitals and adapted modules have a minus super-index at the end of its descriptor.

With this architecture, we are able to handle errors resulting in non-words (through CatSpel); simple orthographic errors resulting in words such as wrong use of apostrophe, or wrong NP-agreement with the morphological error detection module. Other errors related with wrong word sequences are handled with the negative n-gram error detection module. In addition we deal with errors at the morphosyntactic level such as subject-verb disagreement, subcategorization errors, etc. by means of the morphosyntactic error detection module. More complex errors (including certain semantic or pragmatic errors) are detected by means of domain-specific modules (see section 3.1.). We detail the new or adapted modules in the following sections.

2.1. Spell checking during morphological analysis

The first element in Figure 2 is CatSpel (*Cat*alan *Spel*ling). This module actually substitutes the previous morphological mapping module, since it performs a double

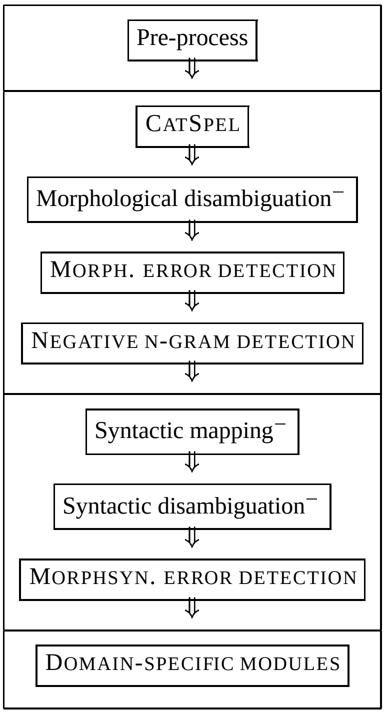
Pre-process
CATSPEL
Morphological disambiguation⁻
MORPH. ERROR DETECTION
NEGATIVE N-GRAM DETECTION
Syntactic mapping⁻
Syntactic disambiguation⁻
MORPHSYN. ERROR DETECTION
DOMAIN-SPECIFIC MODULES

Figure 2: General-purpose NLP-enhanced error detection architecture

task: (i) providing with the readings of those words found in the lexicon, and (ii) providing with correction proposals of those words not found in it. If a non-word is encountered then a set of correct candidates (four edit operations are used: deletion, insertion, substitution and transposition) is found and ranked according to the Minimum Edit Distance algorithm with weights, using the so-called confusion matrixes of (Kernighan et al., 1990).

This module has been implemented in C++, using as a fundamental data structure, a *trie*, in which look-ups are bounded by the length of the targeted non-word, and split-ups and run-ons can be efficiently detected due to the prefix property exhibited by tries. Finally, if two (or more) correct candidates are equally ranked, then some simple heuristics are applied, such as checking whether the suffix of the non-word is related to a particular POS-tag in Catalan ('-ment' for nouns and adverbs, '-ble' for adjectives, and so on).

2.2. Robust and relaxed morphosyntactic parsing

The original architecture of CATCG (see Figure 1) was thought for handling well-formed input. In order to make its CG-based modules compatible and useful for the analysis of ill-formed input we opted for a constraint relaxation.

2.2.1. Relaxation of the morphological disambiguation module

CATCG's morphological disambiguation grammar has been modified in such a way that word sequences that otherwise would imply a wrong or simply a non disambiguation are now analysed as a possible reading. Take a sequence such as the one in Figure 3 *el cadira és maca* [the chair is nice]. In it there is a gender-feature conflict, since the article *el* has a masculine gender and the word *cadira* has feminine gender. Nevertheless, since the word *cadira* is unambiguously a noun, the word *el* will never have a clitic pronoun reading in this context (in Catalan). Thus the system is so defined as to prioritise such a reading, as shown in Figure 3.

```
”<El>”
        “el” EA−−MS
”<cadira>”
        ”cadira” N5−FS
”< és >”
        ”ser” <SSA> VDR3S−
”<maca>”
        ”maco” JQ−−FS
“<$.>”
```

Figure 3: Result of a relaxed morphological disambiguation module. An NP is allowed even if no agreement exists, in certain situations.

2.2.2. Relaxation of the syntactic analyses modules

The relaxation we have used for both the syntactic mapping module and syntactic disambiguation module is in a similar fashion. In the first of these steps, in order to be able to handle with structures that may contain some regular anomalies, e.g. Catalan-Spanish interference errors. Thus the syntactic information assigned takes into account ill-formed word configurations, which result in analyses as the one shown in Figure 4. In this example a sentence such as **el llargandaix es va menjar **a** la mosca* [the lizard ate to the fly], even though the sequence *a la mosca* should never possibly be analysed as a direct object (@CD), it finally has. In order to achieve that, we had to modify the rules that remove the @CD reading for nouns that are preceded by the preposition *a* [to], so that *mosca* [fly] could receive the @CD tag. This information will be useful for the morphosyntactic error detection module, as we will see in the following section.

2.3. Error detection

Error detection is performed in two different phases along the processing streamline. The first one is done right after the morphological disambiguation module. Texts are analysed by two different modules: the morphological error detection module and the negative n-gram detection module. This first module detects NP agreement errors, wrong use of apostrophe and other non-syntactic kinds of errors. Afterwards, the negative n-gram error detection module is run on the text in order to check for certain structural errors such as missing words or word repetitions. Both modules are based on the CG formalism.

The morphological error detection module maps a given error tag to a certain word (reading), if the conditions described in the rule apply. For example, in sequences where there is no NP agreement, as the one shown in Figure 5. The sequence *el bona noi* [the good boy] presents a gender conflict, since the article *el* and the noun *noi* are masculine words, while the word *bona* is a feminine one.

```
”<El>”
        ”el” EA−−MS @DN>
”<llangardaix>”
        ”llangardaix” N5−MS @Subj
”<es>”
        ”se” REER36S @Pr−reflex
”<va>”
        ”anar” <SSPNA> VDR3S− @Vaux>
”<menjar>”
        ”menjar” <SoPoNA> VI− − −− @Vprin
”<a>”
        ”a” P @Advl @<AN
”<la>”
        ”el” EA−−FS @DN>
”<mosca>”
        ”mosca” N5−FS @<P @CD
“<$.>”
```

Figure 4: Analysis provided by the relaxed syntactic modules – *a la mosca* is tagged as direct object (@CD) even though this is an incorrect sequence in normative Catalan

```
“<El>”
        “el” EA−−MS @DN>
“<bona>”
        “bo” JQ−−FS @AN> @:NP−AGR>
“<noi>”
        “noi” N5−MS @Subj @CD
“<$.>”
```

Figure 5: NP-agreement error detected by the morphological error detection module

As for the negative n-gram error detection module, the strategy is to detect based on the fact that they usually imply a completely ill-formed POS-tag sequence. Consider for instance, the impossible sequence *ha una* [have-AUX-VERB a-DET-FEM-SG] which could be the result of forgetting a past-participle like *fet* [done-VERB-PST-PRT]. Up to now only bigrams and trigrams have been used (also implemented using CG-based grammar). They have been collected via a semi-automatic procedure of detection errors in a POS-tagged corpus similar to the one used by (Kveton and Oliva, 2002).

The second phase in error detection takes place after the text has been run through the relaxed syntactic modules –see Section 2.2.. With this module we handle errors in the use of the subjunctive and indicative moods, or other syntactic-based errors such as subject-verb disagreement, and errors resulting from wrong subcategorization patterns (like the one reflected in Figure 4). In the future, a special filtering file will have to be developed in order to prune possible error over-detections, and in order to adapt the correction tool to the needs of different types of user.

3. Customization of a generic error detection architecture

3.1. ALLES: a course for second-language learners

ALLES stands for Advanced Long-distance Language Learning System. This is an EU-funded project the goal of which is to prove that NLP-enhanced error detection contributes to improve feedback and interactive adaptivity in distance language learning environments. The project should result in a short course for intermediate and advanced learners within the business and finance domain in four languages –Catalan, English, German and Spanish, (Schmidt et al., 2004).

Within ALLES in addition to applying the general-purpose corrector for unrestricted text, we have built several specific grammars for each exercise adapted to the type of text and the pedagogical contents that are being taught and evaluated in each learning unit. Therefore, the error detection considers lexical, semantic and pragmatic elements that were not taken into account by the general-purpose error checker. These particular grammars are applied at the end of the process sketched in Figure 2.

This pragmatic checking is achieved by creating templates that correspond to certain communicative functions (or speech acts). For instance, the presence of any of the following linguistic constructions (in Spanish) *pues yo (en tu lugar) (no)* VERB_{cond}, *pero yo (no)* VERB_{cond}, or *tendra(s) que / debera(s)* VERB_{inf} indicates that the speaker is giving advice to the reader/listener. This can be successfully used in domain-restricted situations, which are easy to re-create in language learning environments.

3.2. PrADo: an error checker for native writers

PrADo was a project (ended on December 2003) aiming at developing two grammar checker prototypes for Catalan and Spanish, as well as the establishment of a basis for a future development of style checkers. A solution proposal module on study (but still to be implemented) allows two kinds of corrections: either a set of correction candidates of a word or a display of a warning message. Up to now, only the Catalan prototype has been developed.

4. A toolkit for annotating and exploiting error corpora

The development of wide-coverage error detection tools is clearly favoured by the use of corpora-based error frequency and typology studies –(Granger, 2003) and (L'haire and Faltin, 2003). In this respect, in relation to the above presented projects, we developed a text annotation and exploitation architecture, based on the assembling of previously existing tools. The stages we foresee and for which we provide software facilities are: (i) collecting texts and personal information from corpora providers, (ii) semi-automatic annotation of texts, and (iii) annotated corpora exploitation.

For the first stage we use standard CGI-based HTML forms, and the information compiled refers to aspects such as sex, age, academic and language learning background, etc. For the annotation stage, we provide an MS Word form implemented in Visual Basic that can be used by experts in order to annotate (basically) errors found in text –errors are classified following an adaptation of a classification proposed in (Granger, 2003). A second step in this annotation stage is the fully automatic morphosyntactic analysis of texts performed with CASTCG and CATCG

(Alsina et al., 2002), for Spanish and Catalan texts respectively. All the above information is coded following the TEI P4 guidelines (and labelled with XML tags, (Sperberg-McQueen and Burnard, 2002)).

Regarding corpora exploitation, we compile texts with the Corpus WorkBench (Christ et al., 1999), and use the Corpus Query Processor in order to obtain information such as most frequent errors according to learner level, type of text, etc., as well as real error samples and contexts.

5. Concluding remarks and future work

We have presented a general purpose architecture for error detection tasks and two actual customizations of it for the developing of error detection tools for both native (3.2.) and non-native speakers (3.1.). The current state of development for the ALLES project is far enough in order to start an initial testing phase, which is going to be performed during the April-May 2003.

Nevertheless, we can already advance some of the problems that the system might encounter. Over-detection or overlapping error detection will occur, for which we have started working on the elaboration of an error alarm filtering module, the goal of which is to clarify comments to ALLES users (recall that this is a language learning course, not an error checker for written text).

In addition, with respect to the PrADo project, we have started working on the design of a further refinement of the correction proposals given to users. One of the ways of achieving this is by using the context (e.g., an n-POS-tag window) in order to find the most appropriate candidate for a correction proposal given a certain word sequence.

One other interesting application we foresee for our error detection tools is to use them to semi-automatically label with errors texts provided as error corpora. The idea is to perform a first fully automatic stage of error detection that will be afterwards revised and extended by error annotators (which would normally tag it manually anyway). This all should be integrated in the tool presented in Section 4.

Acknowledgements

This research has been possible thanks to an EU-funded project, ALLES (IST-2001-34246), as well as the project PrADo, funded by the Ministerio de Ciencia y Tecnología from the Spanish government.

6. References

Alsina, Alex, Toni Badia, Gemma Boleda, Stefan Bott, Àngel Gil, Martí Quixal, and Oriol Valentín, 2002. CATCG: a general purpose parsing tool applied. In *Proceedings of 3rd International Conference on Language Resources and Evaluation*, volume III. Las Palmas.

Badia, Toni, Gemma Boleda, Eva Bofias, and Martí Quixal, 2001. A modular architecture for the processing of free text. In *Proceedings of the Workshop on Modular Programming applied to Natural Language Proces*. Iasi, Romania.

Christ, Oliver, Bruno Schulze, and Esther König, 1999. Corpus Query Processor (CQP). User's Manual. Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Stuttgart.

Granger, Sylviane, 2003. Error-tagged Learner Corpora and CALL: A Promising Synergy. *CALICO*, 20-3:465–480.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Atro Anttila, 1995. *Constraint Grammar: a Formalism to Parse Unrestricted Text*. New York: Mouton.

Kernighan, M. D., K. W. Church, and W. A. Gale, 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90*, volume II. Helsinky.

Kveton, Pavel and Karel Oliva, 2002. Detection of errors n part-of-speech tagged corpora by bootstrapping generalized negative n-grams. In *Proceedings of 3rd International Conference on Language Resources and Evaluation*. Las Palmas.

L'haire, Sébastien and Anne Vandeventer Faltin, 2003. Error Diagnosis in the FreeText Project. *CALICO*, 20-3:481–496.

Schmidt, Paul, Toni Badia, Lourdes Díaz, Jorge Fernández, Sandrine Garnier, Martí Quixal, Celia Rico, Ana Ruggia, and Enrique Torrejón, 2004. Advanced Long-distance Language Education System (ALLES): Integrating Language Resources in ICALL. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Lisbon, Portugal.

Sperberg-McQueen, C. M. and Lou Burnard (eds.), 2002. *Guidelines for Text Encoding and Interchange*. Oxford: Humanities Computing Unit, University of Oxford.

Tapanainen, Pasi, 1996. The Constraint Grammar parser CG-2. *Publications of the University of Helsinki*, 27.