CoGrOO – an *OpenOffice* grammar checker

Abstract

This paper describes a Portuguese language grammar checker project, called CoGrOO - Corretor Gramatical para o OpenOffice (Grammar Checker for OpenOffice). Two of its features are highlighted: 1) hybrid architecture, mixing rules and statistics; 2) free software project. This project aims to check grammatical errors such as nominal and verbal agreement, "crase" (preposition "a" + determiner "a" yielding "à"), nominal and verbal government and other common errors in Brazilian Portuguese language. We present some empirical results based on the first version of the grammar checker and the new approach used in the second version.

1. Introduction

OpenOffice.org is a multiplatform and multilingual office suite, and an open source project [1]. This suite has been adopted massively by users and organizations that support free software. However, the lack of a grammar checker is considered as a big problem when we use its word processor component. So, some grammar checker projects in different languages have been developed in order to address this issue [2].

This paper describes a Brazilian Portuguese Language grammar checker project, called *CoGrOO – Corretor Gramatical para OpenOffice* (Grammar Checker for OpenOffice).
In general, written texts are subject to errors such as:

- Spelling errors;
- Grammatical errors: when grammatical rules are not observed, as for example in "Nós vai para casa". ("We *goes* home"). These errors relate to verbal and nominal agreement, for instance.
- Errors of Style: In Portuguese, according to the general rule, the subject precedes the verb, that is, Portuguese is mainly a SVO (Subject-Verb-Object) language. Depending on the context, it is very difficult to understand an inversion. For example: "bonitos eles são" ("pretty they are") is a correct sentence in Portuguese; but in a more formal written style the expected sentence would be "eles são bonitos" ("they are pretty").
- Semantic errors: such errors are strongly context dependent. For example "the truck eats bananas".

The CoGrOO project [3] aims at checking grammatical errors such as nominal and verbal agreement, "crase" (the coalescence of preposition "a" (to) + definite feminine singular determiner "a", yielding "à"), nominal and verbal government, misuse of the adjective "mau" (bad) and the adverb "mal" (badly), among other common errors which can be found in Brazilian Portuguese. In this project, two features are highlighted: - hybrid architecture, mixing rules and statistics; - a free software project.

We released the first version of CoGrOO [4] in 2006 and we are about to release the second version. In this paper, we are explaining the differences between these two versions. In addition, to evaluate the performance of CoGrOO, we compare it to the Brazilian Portuguese language grammar checker in Microsoft Word, called ReGra [5] – these results are also presented.
The remainder of this paper is structured as follows: Section 2 describes the architecture of the checker and the new approach used in second version; in Section 3, some results are discussed and the comparison with ReGra is presented; Section 4, the conclusion, presents the contributions of the implemented approach and some aimed future works.

2. Architecture

The architecture of CoGrOO with its main modules is described in Figure 1. The CoGrOO system is composed of the following modules:

1. Sentence Boundary Detector: receives a text and split it into sentences;
2. Tokenizer: receives a sentence and split it into words and punctuation marks;
3. Name finder: receives the sentence tokens and identifies the potential proper nouns;
4. Part-of-Speech Tagger: receives a sentence and assigns morphological tags to its lexical items.
5. Chunker: receives a tagged text and finds small noun phrases (NP) and small verbal phrases (VP).
6. Subject-Verb Finder: receives a tagged sentence with NPs and VPs and searches for the subject. If it is found, it marks an NP as a subject of a VP.
7. Grammar Error Detector: this module looks for grammar errors in the input sentence. It is activated after main steps of sentence analysis.

An important linguistic resource used in CoGrOO project is the CETENFOLHA corpus [6]. CETENFOLHA is a Brazilian Portuguese morpho-syntactic annotated corpus, based on journalistic essays. It is not colloquial, generally written in third person.

Basically, CoGrOO version 1 and 2 follows the same architecture. In CoGrOO 1, we used CETENFOLHA corpus to train the tagger, chunker and Subject-Verb Finder. All these three modules were hand-coded, full of heuristics, and they do not follow any particular machine learning algorithm. It resulted in a code, written in Perl programming language, that is very difficult to maintain. In CoGrOO 2 [7], we used OpenNLP, a Natural Language Processing (NLP) library, in order to create a more consistent and robust software. From its site, we extracted a small description of the OpenNLP [8]:

> "*Its primary role is to encourage and facilitate the collaboration of researchers and developers on [NLP] projects.*
> *OpenNLP also hosts a variety of java-based NLP tools which perform sentence detection, tokenization, pos-tagging, chunking and parsing, named-entity detection, and coreference using the OpenNLP Maxent machine learning package."*

The NLP tools in OpenNLP are created for English, but it was not difficult to adapt the tagger and the chunker for Portuguese. We had to make some changes in the English features and train it using the CETENFOLHA corpus.

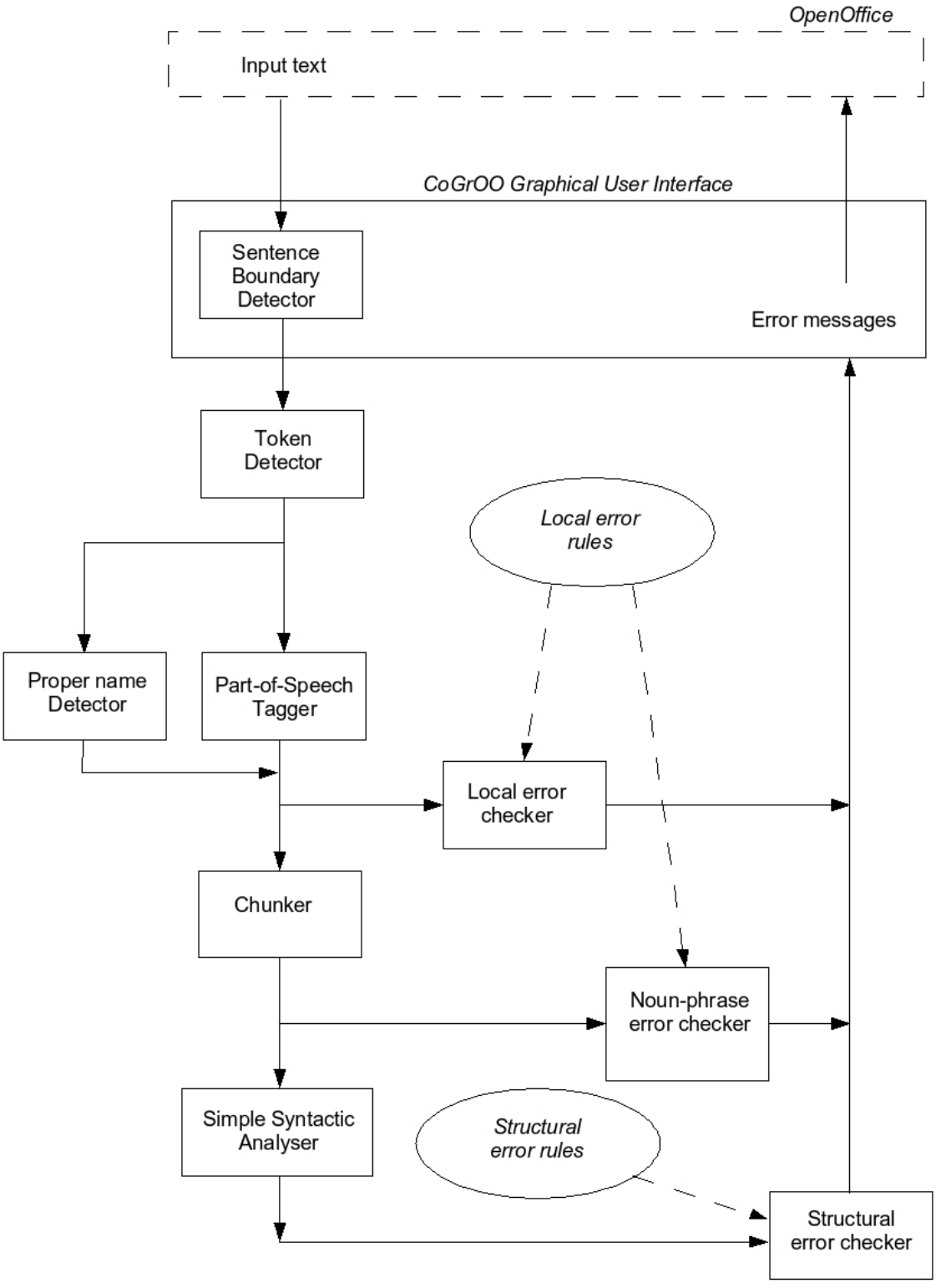

Figure 1 – CoGrOO system architecture

In the following items, we describe each system module with more details and how they accomplish the error-checking task.

2.1 Sentence Boundary Detector

The sentence boundary detector is responsible to identify the sentences based on ponctuation mark. It checks, for instance, if a period mark ends a sentence or if it is part of an abbreviation. Similar consideration must be done to other marks that can split Portuguese sentences, like exclamation, question and quotation marks, semicolon, brackets, dashes, and ellipsis.

Period, exclamation, question, semicolon and ellipsis are considered absolute sentence splitters. Brackets, dashes and quotation marks are only considered if the sentence contains a finite verb. Example: *o menino disse: "eu quero" e depois arrancou o brinquedo do irmaozinho. (the boy said: "I want" and then, took the toy from his little brother").* In this case, the quotation marks highlights a sentence and a clue is that it contains a finite verb.

In CoGrOO 1, the steps adopted to detect sentences were:

1. Identify punctuation marks in the text;
2. Perform a match with some masks to check for constructions that use punctuation marks, but don't define sentences, like e-mail and Internet addresses;
3. If it is a period mark, check if it is part of an abbreviation notation; exclamation, question, semicolon and ellipsis are considered sentence splitters;
4. Brackets, dashes and quotation mark are considered sentences splitter only if there is at least one finite verb.

CoGrOO 2 is much similar; however, by using OpenNLP maximum entropy algorithms, it is possible to evaluate more features (example: neighbor words around the poncutation mark) and estimate the relevance of them . The training is done by sending sentences to the NameFinder module which extracts their features and assembles a database that will be used in the maximum entropy algorithm.

Example of how the Sentence Boundary Detector module works on the input text:

Input: “O sr. Mendonça chegou. Entrego-lhe os documentos?”

Output: “{O sr. Mendonça chegou .} {Entrego-lhe os documentos ?}”

2.2 Tokenizer

Sentences are composed of blocks of text, named tokens. The tokens usually represent words and other marks, like punctuation marks.

Generally, a sequence of tokens is something like: (token1, space, token2, space, etc.), but sometimes there is no space between tokens, mainly when a ponctuation mark follows a word.

The tokenizer must detect the tokens.

CoGrOO 1 tokenizer adopted a state machine to recognize tokens.

Basically:

1. White spaces split tokens;
2. Sequence of alphanumeric are tokens;
3. Punctuation marks are tokens;
4. A sequence of alphanumeric, followed by a period mark, that exists in the abbreviation dictionary, is considered one token;
5. A sequence of characters, that matches with constructions like e-mail or Internet addresses, is considered one token.

CoGrOO 2 used the Tokenizer module of OpenNLP. It works in 2 steps:

1) it splits the sentence in white spaces. Example: "Entrego-lhe os documentos?" would yield the strings: [Entrego-lhe] [os] [documentos?].

2) for each string, it checks its features and compares to the training data. Some features are:

1. The end of the string is a potential token splitter (not alphanumeric) ?
2. The end of the string is a period mark and it is not part of an known abbreviation ?
3. The string matchs some known construction, like an e-mail address ?
4. The character before the punctuation mark is a capitalized letter?

The training is done by sentences of the annotated corpus. Through them it is possible to create a database with the answers to the features. The maximum entropy algorithm use it to weight when ponctuation mark will be a token or not.

Example:

Input: "[O sr. Mendonça chegou.] [Entrego-lhe os documentos?]"

Output: "{ [O] [sr.] [Mendonça] [chegou] [.] } { [[Entrego] [-lhe] [os] [documentos] [?] }"

2.3 Name finder

The Brazilian Portuguese corpus (CETENFOLHA) arranges proper nouns in just one token. So, to use the corpus as a trainning data to the tagger, we should first identify and group proper nouns.

CoGrOO 1 used a state machine to recognize words with some characteristics and then group them together. They are:

1. Fist letter is capitalized;
2. Token that is tagged as proper noun in the dictionary or is an unknown word;
3. In Portuguese some names contains prepositions, so a preposition in between two proper nouns should be part of the noun. Example: "Joao da Silva".

CoGrOO 2 used the Name Finder module of OpenNLP. The training process is done by sending to the module sentences with the proper nouns marked with tags: "I think <START>Jane Doe<END> is coming." and the module will search the characteristics of the occurrence and then evaluate the importance of each characteristic to determine if it should be considered when evaluating new texts. These characteristics are:

1. Capitalization;
2. Tokens classified as proper nouns in the dictionary;
3. Unknown word (not in the dictionary)
4. Occurrence of prepositions in between nouns.

For this module a performance evaluation was done. The best mark was 93% of assurance, training with 50,000 sentences.

Example:

Input: "{ [As] [crianças] [se] [divertiram] [na] [Casa] [da] [Boa] [Esperança] [.] }" ("The children had fun in the House of the Good Hope").

Output: "{ [As] [crianças] [se] [divertiram] [na] <START> [Casa] [da] [Boa] [Esperança] <END> [.] }"

2.4. Tagger

This module assigns a morphological tag for each word of the sentence.

The tagger follows the steps:

1. assigns all the possible tags to each word of the sentence.
2. defines the most probable tag for each word of the sentence, by inspecting its context.

Step (1) uses a dictionary that assigns possible tags for each word. This dictionary was generated by processing the CETENFOLHA annotated corpus. The process is in the following way: we counted how many times a word W was tagged as T. The most probable tag to word W is the one which appeared more times in the corpus. We also make use of a suffixes directory to handle words missing from the dictionary. The last 3 letters are searched for, and, in cases where the suffix cannot be found, we simply assigned the tag "singular noun" to the word, as this is the most recurrent tag in the Portuguese language.

In the second step we have to choose just one tag to each word based on an algorithm similar to Brill's tagger [9]. For each word, the algorithm will replace the most probable tag by another tag, by inspecting the neighborhood, i.e., a sequence of of three tags (tag-trigrams). We have to choose a tag that most resemble the tag-trigrams from CETENFOLHA. We extracted the tag-trigrams from CETENFOLHA. We decided to use just the 80% more frequent trigrams. Just small patterns respond for the majority of the trigrams (Zipf's law).

The tagger in CoGrOO has a precision rate of 95%. It was built specially for this project, since we did not have access to an open source Brazilian Portuguese tagger.

But we had some problems in this first version: we extracted trigrams from the CETENFOLHA corpus, similar to [10]. We assigned weights to the trigrams based on their occurrences. For each word be tagged, there are 3 trigrams that covers it. We created a heuristic in order to choose how the word will be tagged based on the trigrams weights. This heuristic required a manual adjustment and we found that this process was very difficult to maintain. Even though, the tagger had a good performance similar to those for the English language [9].

In CoGrOO 2, tagger was based on the English tagger that comes with OpenNLP which is according to [11]. In order to adapt the OpenNLP tagger to the Portuguese language, we made changes in some features and we obtain similar performance to the previous version. With this approach we do not need to make manual adjustments yielding a more stable source code. This result will be important when adapting CoGrOO to other languages.

The performance results showed up to 96.5% of assurance training with 50,000 sentences.

2.5. Chunker

It uses the part-of-speech information provided by the tagger and employs a finite state machine (CoGrOO version 1) to detect boundaries of syntactic groups. The chunker leaves all previously added information in the text and creates structural elements which include words of the chunk. For example:

<NP>The man</NP> from <NP>São Paulo</NP><VP>was</VP> <NP>happy</NP>.

Currently it is capable of recognizing boundaries of simple noun and verb groups. Noun groups do not include prepositional or clausal post-modifiers.

The finite state machine was automatically got from the CETENFOLHA corpus. The NP and VP sequence of tags was extracted from the corpus and we assembled the finite state machine from them.

In CoGrOO Version 2, the chunker was based on the English version that comes with Open NLP that is based on the Maximum Entropy algorithm. The CETENFOLHA corpus was used to train the chunker but it had to be modified. For each word of the corpus, we changed its syntactic information to one the tags: BNP - word that begins a NounPhrase, INP - a word inside a NounPhrase, BNP - a word that begins a Verbal Phrase, IVP - a

word inside a Verbal Phrase or NULL (does not belong to a NP or VP such as a preposition). These tags and their use in order to train the chunker are explained in [12].

2.6. Subject-Verb Finder

The purpose of this module is to detect the subject (a NP) and a verb (a VP) relationship. The output of the chunker and the tagger is submitted to this module. It searches for a NP-VP pattern, i.e., a pattern of tags, NPs and VPs. For example:
(!, NP, VP)
where
! = Begin of sentence.
If we find a NP beginning a sentence followed by a VP then the pattern is found and NP will be assigned as the subject of VP.
These patterns were automatically extracted from CETENFOLHA corpus. Whenever the syntactic tag SUBJ appears in the corpus, a NP-VP pattern is extracted. For example, lets us suppose that the sentence "o menino caiu no buraco" (the boy fell in the hole) were in the corpus. We would have found the following tags assigned to its words:
" *! NP1(o menino/the boy) VP(caiu/fell) PREP(em/in) NP2(o buraco/the hole)"*
and through the syntactic tag SUBJ that is in the corpus, NP1 is assigned as the subject of VP. Thus the NP-VP pattern:
(!, NP, VP)
is automatically extracted from the *corpus*.
However, it may happen that some sentences that match this pattern do not have NP as subject of VP. Thus we count the number of sentences (S) where this pattern appears in the corpus and the number that, in this case, NP is subject of VP. In this case, it holds in 85% of S. If this number is higher than a manually given threshold then we will use this NP-VP pattern to extract subjects from new sentences.
These patterns will be gathered in a finite state machine in order to detect the subject of a given sentence. It may happen, that no pattern matches and a subject is not found in the sentence.
The first and second version of CoGrOO uses the same approach. We did not find any module in OpenNLP similar to this module.

2.7. Grammar Errors Detector

This module detects possible errors in the input sentence through the use of hand-written error rules. An error rule is just a pattern that is searched in the input sentence. These rules are based on common grammar errors in Brazilian Portuguese.
A typical example of error detection by this module is the use of the “crase” – which is the contraction between the preposition “a” (“to”) and the singular feminine definite article “a” (“the”) – before masculine words or verbs. So, an example of error rule is:
(“**a”, “a”, verb)**
This rule represents a “à” (“a” with crase) followed by a verb. In this case, “à” is represented by the sequence (“a”,“a”) because in Portuguese à = (“a”,“a”) and CoGrOO uses the last representation.
The module input is the sentence with data provided by the former modules: sequence of tokens, tags, phrases and subject-verbs marks. The output is a data structure which marks the error in the sentence. It is composed of an error message, the wrong sentence and suggestions on how to deal with the grammar error.
We activate the Grammar Error Detector Module in three different times: after POS-tagging, after chunking and after shallow parsing. After POS-tagging, we have the local rules that are applied to a short sequence of words

and tags. For the treatment of more complex errors like verbal agreement errors, we apply structural error rules in the output of Chunker and Subject-Verb Finder modules.
In the first version, we had a problem: as the rule syntax wasn't automatically checked, we had a lot of errors in the hand-written rules that were difficult to detect. At CoGrOO 2, rules are defined in a XML schema. We also used the Altova XMLSpy [13] software, evaluation version in order to specify the syntax of the rules and detect syntax errors. This software offers a visual interface that makes easier this work.

3. Results

We released CoGrOO 1 in 2006. Despite some problems, it was installed in some large companies such as Itaipu Binacional, the company that runs the largest operational hydroelectric power plant in the world, and Companhia do Metropolitano de São Paulo, a public transport company, and known popularly as Metrô-SP. As CoGrOO is free software, we do not know how many computers run CoGrOO, but only Itaipu has adopted CoGrOO in 70% of its computer park (about 2,100 computers) [14].
In order to have a better evaluation of CoGrOO 1, we created the Metrô corpus [15] from the site of Metrô-SP [16]. We used this corpus to compare CoGrOO to ReGra, a grammar and style checker for Brazilian Portuguese language [5] used in Microsoft Office 2000. It is a corpus with 16,536 words and about 800 sentences; it is consisted of small pieces of texts, each one with about 4 paragraphs.
In this corpus, we have well written documents, probably analyzed only by human revisers. However, it was possible to detect errors in these texts. The reason for this is no automatic revision tool (like CoGrOO or ReGra) was used, probably.
In this experiment, we used two parameters to evaluate the system performance:

- True positives: grammar errors correctly accused.
- False positives: accused grammar errors that do not exist.

In order to check these parameters, a human expert, a linguist, analyzed the corpus and detected 51 grammar errors. Moreover, ReGra detected 10 stylistic errors correctly, but 2 false positives were pointed too. Stylistic errors are concerned to typographical conventions, for example: - two or more consecutive spaces between words; - a space before a comma. Unlike Regra, CoGrOO system is purely a grammar checker system. Table 1 shows the results of these experiments.

GRAMMAR ERRORS	CoGrOO	ReGra
True positives	14	15
False positives	10	36

Table 1: Grammar errors detected by CoGrOO and ReGra

CoGrOO and ReGra detected 7 common true positives (crase, agreement and punctuation errors). CoGrOO detected 8 true positives that Regra didn't detect, for instance, some nominal government errors. By the other

hand, Regra system detected 7 true positives that were ignored by CoGrOO system, like some specific punctuation and agreement errors.
In a grammar checker system, each new rule can increase the amount of true positives detected, but it can also increase the amount of false positives, which is not desirable. By analyzing the messages emitted by ReGra, and also the correction suggestions, we can observe that it implements more rules than CoGrOO. Therefore, we achieved a better ratio between true and false positives than Regra in the Metrô corpus. CoGrOO has around 100 rules in its base.
For each grammar checker rule, we counted the number of true and false positives that it yielded when applied to the Metrô corpus. Rules with a low ratio were discarded or revised.
The stylistic errors module, yet to be done, will improve this ratio even more. We must confess that our rule set has been changing during Metrô corpus analysis due to that procedure of scoring rules, in order to select the best set of rules.
Until now, we've been working with Metrô corpus, but we must apply CoGrOO to others corpora in order to evaluate our grammar checker. We also intend to repeat and extend these evaluations with CoGrOO 2.
Although we got a better ratio between true and false positives, we know that ReGra is better than CoGrOO, because it has a better Portuguese parser, a greater number of error rules and it also contains a stylistic error module.

4. Conclusions

CoGrOO project follows a hybrid architecture, mixing rules (symbolic processing) and statistics – machine learning based on annotated corpus training, using Maximum Entropy Algorithm. The approach used in the Subject-Verb Finder (the search for syntactic patterns on annotated corpus) is innovative; we didn't find similar approach related to subject-verb detection in the literature.
We intend to implement a stylistic error module because these rules are easy to write and it is difficult to generate false positives.
CoGrOO's road map includes its porting to other languages – our initial targets are English and Spanish. This task is not so complicated because OpenNLP is not attached to a particular language.

5. Acknowledgements

We would like to thank to CoGrOO team members: Bruno Cesar Brito Sant'Anna, Gabriel Bakiewicz, Diogo Abreu Meyer Pires, Fábio Wang Gusukuma, Marcelo Suzumura, William Daniel Colen M. Silva and the linguist Sueli Caramello Uliano.
This research was supported by Finep (Financiadora de Estudos e Projetos), Public Call MCT/FINEP/ FINEP-01/2003. We also would like to thank the FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) for hosting our source code (CVS) and site.

6. References

[1] OpenOffice site (http://www.openoffice.org, last access: dec. 02nd, 2006).

[2] Lingucomponent Sub-Project: Grammar Checking (http://lingucomponent.openoffice.org/grammar.htm, last access: dec. 02nd, 2006).

[3] J. Kinoshita, L.N. Salvador, C.E.D. Menezes, "CoGrOO – Um Corretor Gramatical para a língua portuguesa, acoplável ao OpenOffice", In *Proceedings* of XXXI Latin American Informatics Conference, CLEI 2005, Cali, Colombia, 2005.

[4] J. Kinoshita, L.N. Salvador, C.E.D. Menezes, "CoGrOO: a Brazilian-Portuguese Grammar Checker based on the CETENFOLHA Corpus", In *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006*, Genoa, Italy, 2006.

[5] M.G.V. Nunes, O.N. Oliveira Jr., "O processo de desenvolvimento do Revisor Gramatical ReGra", *Proceedings of XXVII SEMISH (XX Congresso Nacional da Sociedade Brasileira de Computação)*, Volume 1, p.6 (abstract). Full paper on CD-Rom version, PUC-PR, Curitiba, Brazil, 2000.

[6] Linguateca, "CETENFolha", Brazilian-Portuguese annotated *corpus* (http://www.linguateca.pt/CETENFolha, last access: dec. 02nd, 2006).

[7] W.D.C.M. Silva, M. Suzumura, F.W. Gusukuma, D.A.M. PIRES, "Corretor Gramatical Acoplável ao OpenOffice.org - CoGrOO 2.0", Bachelor on Electric and Computing Engineering end-of-course Monograph, Polytechnic School, University of São Paulo, Brazil, 2006.

[8] OpenNLP, open-source framework to develop natural language applications (http://opennlp.sourceforge.net, last access: dec. 02nd, 2006).

[9] E. Brill, "A Simple Rule-Based Part Of Speech Tagger", *Proceedings of ANLP-92, 3rd Conference of Applied Natural Language Processing*, Trento, Italy, 1992.

[10] C.E.D. Menezes, "Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos", Master Degree Thesis, University of São Paulo, Brazil, 2000.

[11] A. Ratnaparkhi, "Maximum Entropy Models for Natural Language Ambiguity Resolution", Ph.D. Dissertation, University of Pennsylvania, July 1998.

[12] L.A. Ramshaw, M.P Marcus, "Text chunking using transformation-based learning", *In Proceedings of the Third ACL Workshop on Very Large Corpora*, Association for Computational Linguistics, 1995.

[13] Altova XMLSpy - XML editor for modeling, editing, transforming and debugging XML technologies (http://www.altova.com/products/xmlspy/xml_editor.html, last access: dec. 02nd, 2006).

[14] M.S. Martins, personal communication of Marcos Siríaco Martins to William Colen Silva, during III Conferência Latinoamericana de Software Livre (Latinoware), november 16-17, 2006.

[15] S.C. Uliano, C.E.D. Menezes, F.W. Gusukuma, "Uma análise do CoGrOO, um Corretor Gramatical acoplável ao OpenOffice" (http://cogroo.incubadora.fapesp.br/portal/down/Doc/analise, last access: dec. 02nd, 2006).

[16] Metrô, institucional site of "Companhia do Metropolitano de São Paulo", available at http://www.metro.sp.gov.br, last access: dec. 02nd, 2006.