Selva: A New Syntactic Parser for Portuguese

Sheila de Almeida, Ariadne Carvalho, Lucien Fantin, and Jorge Stolfi

Institute of Computing, State University of Campinas
Cx. Postal 6176, 13084-971 Campinas (SP), Brazil
{sheila.almeida,ariadne,lucien.fantin,stolfi}@ic.unicamp.br

Abstract. We describe an ongoing project whose aim is to build a parser for Brazilian Portuguese, Selva, which can be used as a basis for subsequent research in natural language processing, such as automatic translation and ellipsis and anaphora resolution. The parser is meant to handle arbitrary prose texts in unrestricted domains, including the full range of coordination and subordination constructs. The parser operates separately on each sentence, and outputs all its syntactically valid derivation trees.

1 Introduction

Here we describe **Selva**, a new syntactic parser for Brazilian Portuguese, designed to deal with arbitrary prose text, without domain or context restrictions, and allowing the full range of coordination and subordination constructs. The parser operates separately on each sentence (as delimited by periods or other full stops), and outputs all its syntactically valid parsings. We consider only plain running text, excluding styles with special structures like poetry and headlines.

The main obstacles to the robust parsing of real-world text are the enormous number of linguistic constructions that must be handled, and the numerous syntactic ambiguities. The latter can only be resolved by semantic analysis, which requires an intelligent understanding of the whole text and of its origins—which in turn requires an impossibly a vast and complex world model, and powerful logical/probabilistic inference methods. This difficulty is especially serious for unrestricted text, where even green ideas may sleep furiously on occasion.

A parser that uses only syntactic criteria, such as word categories and word order, will be unable to choose the correct parsing among all possible parsings for the same sentence; it will have to guess, based on some heuristic or statistical rules. Considering that a typical sentence of moderate length may require dozens of such choices, the chances of making the right guess at evey one will be very small. The only way that a parser can approach 100% success rate is by listing all, or nearly all, syntactically valid parsings for each sentence. Even though an n-word input may admit thousands of such parsings, they are only a tiny fraction of all possible trees with n leaves. We concluded that a tool that found all valid parsings would be very useful for language processing, e.g. as a front-end syntactic filter for a restricted-domain semantic analyzer.

Having chosen to generate all possible parsings, we found it best to avoid many traditional rules which are inspired on semantic criteria — such as transitivity — as being too unreliable. We also decided against statistically-based filters, since rule usage probabilities are strongly dependent on semantical context. The Selva parser assumes that the input is syntactically correct, and makes no effort to reject ungrammatical sentences. Finally, we do not try to flag meta-syntactic constructs such as passive voice or cleft predicatives (`Ela é quem fez o bolo`).

On the other hand, in order to keep the size of the output within tolerable limits, and avoid generating invalid parsings for correct sentences, we found it necessary to enforce certain syntactic restrictions, such as person and gender agreement, and to exclude some phrase structures which, although formally valid, are too rare to be worth considering. For instance, we do not recognize clauses with untopicalized object-subject-verb order (as in `O carro Maria comprou`)— even though they occasionally occur, even in prose.

2 Related Work

There are surprisingly few projects envolving syntactic analysis of Portuguese [10]. Moreover, some of them are commercial projects unavailable for research, while others are developed for very limited domains. We are aware of only two accesible parsers that can be compared to ours: Curupira and VISL.

The Curupira parser of Martins, Hasegawa, and Nunes [4] was developed as part of ReGra, a commercial grammar checker [5]. Like our parser, Curupira assumes that the sentence is correct and generates all possible parse trees. The parser is still under development, and its source has not been released.

The VISL parser was developed by a team led by Eckhard Bick within the Visual Interactive Syntax Learning project [1]. Apparently the source code is not available, but the parser itself can be used for demonstrations through the Internet. The parser is very robust and produces generally good results; however, it only provides one parsing for each sentence.

3 The Grammar

We encode the syntax by a context-free grammar with markers. The syntax is loosely based on standard Portuguese grammars [7, 9, 6, 2], but we were forced to deviate from them in many points, chiefly due to the absolute lack of semantic information. (Even grammarians who take pains to separate syntax from semantics, like Perini, occasionally define syntactic categories by semantic tests.) Another reason was the need to handle complex coordination phenomena which are ignored by most grammars.

Categories and Functions. As usual, we assign two labels to each phrase in a sentence: its syntactic *category*, and its syntactic role or *function* within the immediately containing unit. The most important sytactic categories are *sentence*, *clause*, *verb*, *noun*, *adverb*, *adjective*, and *prepositive* (prepositional phrase). (Other categories like *preposition* and *article* occur as constituents of these.) These categories are not disjoint; for example, in `eu quero correr`, the word `correr` is classified as a verb at a lower level of the parse tree, as a clause at a higher level, and as a noun further up.

The top-level syntactic category is the sentence, which we define as a sequence of words delimited by strong punctuation (full stop, colon, semicolon, question, or exclamation marks). The clause category applies to sentences, or parts thereof, that consist of a verb and all its syntactically bound operands and modifiers, including subordinate clauses. (Most sentences are clauses, but occasionally one finds verb-less sentences such as in `Ele demorou. Bastante.`.)

Markers. Each category is further subdivided into sub-categories, characterized by certain parameters (*markers*) which can assume a finite range of values. Thus, for example, the noun class comprises twelve main sub-categories, characterized by three markers: *gender* (MAS or FEM), *number* (SIN or PLU), and *person* (1, 2, or 3). These markers are used to implement agreement constraints, and are strictly syntactical, not semantical; and many phrases, such as `lápis` or `que saiu`, are ambiguous with respect to them. Nouns generally have person 3, except for pronoun-based ones such as `tu` or `apenas eu e você`.

Adjective phrases have the same sub-categories as nouns. Here the person marker is significant only for adjectives built from subordinate clauses: `que fomos contratados` has person 1, `que foram contratados` has person 3. Adverbs and prepositives do not have gender, person, or number markers.

Verb phrases are sub-classified by four main markers: *mood*, *person*, *number*, and *gender*. The mood can have seven possible values, such as *indicative* (IND), *subjunctive* (SUB), *past participle* (PAR), etc.. Other markers identify verb phrases which include oblique (clitic) pronouns. We found no compelling reason to mark verb forms for tense (past, present, etc.), or to distinguish copular verbs like `ser` from ordinary ones. We also did not find it useful to mark verbs for transitivity, since every transitive verb may be used with elided object, and many supposedly intransitive verbs can have special meanings which are transitive.

Clauses have several markers, such as the mood of the clause's main verb. An important one is *incompleteness*, which characterizes clauses that have an elided noun constituent, like `Maria comprou []` ou `eu disse que [] saíram`. Such clauses are used to build adjective phrases (see below).

3.1 Clause Structure

A normal clause consists of a single verb phrase, surrounded by optional noun, adverb, adjective, or prepositive phrases, possibly with some punctuation. Each of these constituents has a function in the clause, which can be *topic* (T), *subject* (S), *object* (O), *object complement* (C), and *clause modifier* (M).

There may be one or more topics T at the beginning of the clause. Each topic is a noun, adverb, adjective, or prepositive phrase, or an *expletive*, which includes vocatives and interjections, always followed by a comma, such as `O gato, ninguém o viu`, or `Branco, vermelho, qualquer cor serve`.

Other than topics and appositives (see below), at most three constituents of a clause can be noun phrases; these are assigned the functions S, O and C. The object complement C may also be a non-clausal adjective. The complement occurs, for example, in the sentences `ele nomeou Pedro ministro` (noun) `ele deixou os eleitores frustrados`.

There may be any number of clause modifiers M, inserted before, between, or after the S, V, O, and C constituents. Each M may be either an adverb, an adjective, or a prepositive. The subject S must agree with the verb in person and number, but there is no such constraint between them and the object O. When the complement C is an adjective, it must agree with the object O in gender and number. Any clause modifiers M which are adjectives must agree with the subject. (Note that these constraints often allow us to distinguish an adjectival C from an adjectival M, as the word `frustrados` in the previous example.)

If we ignore T and M phrases, we have 24 potential orders for the constituents S, V, O, and C; which become 49 considering that S, O, and C may be absent. Although all these combinations are theoretically valid and occur in special contexts, we found that almost all sentences in our corpus used only the following alternatives: $SVOC$, $SVCO$, $SOVC$, SVO, SVC, SOV, VSO, VS—and their variants without the subject S, e.g. `Achei o livro chato` (VOC). Moreover, some of these orders are possible only under certain constraints; for instance, $SVCO$ is not allowed if the complement C is a noun: `*o presidente nomeou ministro Pedro`.

Noun Phrases. The typical noun phrase has the structure $M^* DQ^* HQ^* M^*$. All parts may be omitted (under some conditions), except the *head word* H, which is either a noun, adjective, prepositive, or pronoun. The *qualifiers* Q may be adjectives or (after the head) prepositives. The *determiner* D may be an article or demonstrative pronoun; the *modifiers* (M) may be adverbs or prepositives. Cf. the example `[somente]`$_M$ `[o]`$_D$ `[maior]`$_Q$ `[livro]`$_H$ `[de exercícios]`$_Q$ `[verde]`$_Q$ `[que comprei]`$_Q$. A noun phrase may also be formed from a subordinate clause, in a number of ways, e.g. `[dançar samba] é bom`, `vi [os cavalos bebendo água]`.

A major source of structural ambiguity is the fact that a single adverb, adjective, or prepositive may often be parsed as a constituent of several categories at several levels. The noun phrase `a caixa de madeira sem a alça da tampa` admits many different tree structures besides the semantically correct one `[[a]`$_D$ `[caixa]`$_H$ `[de madeira]`$_Q$ `[sem [[a]`$_D$ `[alça]`$_H$ `[da tampa]`$_Q$`]]`$_Q$`]`.

Adjectives, Adverbs, and Prepositives. And adjective is either a single word (the *headword*, H), or one of several constructs with subordinate clauses, such as `que`+clause (`que Maria comprou`) or `cujo`+noun+clause (`cujo carro Maria comprou`). A prepositive consists of a preposition (P) followed by a noun (the *body* B)—which may be a subordinate clause, e.g. `máquina [de [fazer` `macarrão]]`, `viaje [sem [sair de casa]]`. Non-clausal adjectives and prepositives may be modified by adverbials (M), as in `bem branco` or `muito de confiança`.

The subordinate of `que`-adjectives must be an incomplete clause (see below) as in `o carro [que [Maria comprou []]]` and `as pessoas [que [eu disse que [] saíram]]`. The second example, where `pessoas` must agree in person and number with `saíram`, shows that markers of clause incompleteness and noun type must sometimes be propagated through several levels of the parse tree.

Adverbs can be either a single word or one of several subordinate constructions, such as `quando eu nasci`, `se não chover`; or arbitrary phrases isolated from the sentence by parentheses, dashes, or paired commas — which include vocatives, expletives, etc.. Adverbs too can be modified by adverbs and prepositives. Many simple adjectives, such as `rápido`, can also be classified as adverbs, and there seems to be no reliable sintactic criterion to distinguish a prepositive used as an adjective (qualifying nouns) or as adverb, except, occasionally, by agreement constraints or similar contextual information.

Indirect Object. We found it impossible to distinguish between a clause modifier and an "indirect object" (except a pronominal one). The distinction is traditionally depends on the verb being tagged in the lexicon as "indirect transitive." However, this approach fails too often to be of much use. In fact, there are examples which are inherently ambiguous, such as `O menino gosta de verdade` — `de verdade` may be either *what* the boy likes, or *how much* the boy likes it. We also did not find it helpful to mark verbs with its "usual" prepositions (*regência*), since it seems that any preposition can be used to modify any verb. We were forced to conclude that the concept of "indirect object" is largely a matter of semantics, not syntax. Therefore, we parse prepositionals like `de verdade`, and the verb-attached weak "indirect" pronouns like `lhe` and `se`, as clause modifiers.

Compound Verbs. The traditional parsing of clauses like `ele vai fazer isso` (or `ela tinha feito isso`) classifies `vai fazer` (resp. `tinha feito`) as a "compound verb", and labels `isso` as the object of the top-level clause. We found this approach problematic in view of coordinations like `ele vai fazer isto e pensar naquilo`, or inserted terms like, `ele vai apenas me encontrar`.

Threfore, we chose to parse the main verb in such constructions, together with any direct object and modifiers, as a subordinate noun phrase which is the direct object of the auxiliary verb. Under this view, we must interpret the auxiliary `ir` as a special sense of the verb, which is transitive—the object being the action that is going to happen. (However, we still have special tratment for clauses where a weak pronoun—direct object or clause modifier—belonging to the main verb is displaced in front of the auxiliary, as in `ele me quer ver`.)

3.2 Coordination

Coordination is a major difficulty in syntactic analysis. In typical text, besides coordination of whole clauses or major phrase categories (nouns, adjectives, verbs, and adverbs), one finds problematic examples such as `Tenho camisas com e` sem gola, Quero este mas não aquele livro. Coordination may also occur between multi-phrase sequences, as in ele está proibido de entrar em ou sair de casa, Maria comprou o carro e vendeu a casa simultaneamente, Ele nomeou Pedro diretor e Maria assistente, etc..

A popular solution is to view such constructs as coordination between clauses, with many elided parts: for instance, Maria compra e vende imóveis is parsed as [Maria compra []] e [[] vende imóveis], where [] stands for an omitted constituent. However, this interpretation is problematic because it requires forward anaphoric references (*cataphora*) and the elision of parts which should not be elided, and it also breaks the semantics of adverbials like mutuamente and simultaneamente. Threfore, coordination must be handled as a general phenomenon that can operate on two or more phrases of almost any category X, to produce a phrase of the same category X [8]; or even on groups of phrases which do not form a single syntactic unit, e.g. between two subject-verb pairs as in João vendeu e Maria comprou o carro.

4 The Pre-Processor

Before each input clause is submitted to the parser, it goes through a pre-processor, which breaks the text into words, obtains the word categories (from a large dictionary, which can be supplemented by the user, and some simple heuristics for numbers and proper names) and finally turns it into a list of Prolog clauses.

Contractions like dele and lhos are flagged as such in the main dictionary, and are expanded by the pre-processor. Since some some contractions are ambiguous, the result is no longer a *sequence* of tagged words but rather a branching directed graph. For instance, the clause vamos nos encontrar becomes:

$$\bullet_0 \to \texttt{vamos} \to \bullet_1 \begin{array}{ccc} \nearrow & \texttt{nos} & \searrow \\ \searrow & \texttt{em} \to \bullet_2 \to \texttt{os} & \nearrow \end{array} \bullet_3 \to \texttt{encontrar} \to \bullet_4$$

5 The Parser

The Selva parser is implemented in Prolog [3]. Rather than using the DCG parsing facilities built into Prolog, we use a separate translator to map the source grammar into plain Prolog rules. The translator adds an interval constraint to each term of each rule, and reorders the terms so as to avoid infinite recursion. Even though the parser depends on Prolog's top-down search with backtracking, the extra parameters and rule reordering give it some of the robustness and effciency expected from bottom-up parsers. A typical rule in the source grammar

$$\begin{array}{l} \texttt{sentence}(\texttt{MOOD}, ...) \to \\ \quad \texttt{subject}(\texttt{PERSON}, \_, \texttt{NUMBER}), \\ \quad \texttt{*verb}(\texttt{MOOD}, \texttt{PERSON}, \texttt{NUMBER}), \\ \quad \texttt{object}(\_, \_, \_). \end{array} \tag{1}$$

gets translated into the following Prolog code

```
sentence(NI, NS, INF, SUP, MOOD, ..., T) :-
verb(N1, N2, INF, SUP, MOOD, PERSON, NUMBER, T2),
subject(NI, N1, INF, SUP, PERSON, _, NUMBER, T1),
object(N2, NS, INF, SUP, _, _, _, T3),
buildtree('sentence_1', T, [T1, T2, T3]),
```

Each predicate matches directed paths in the input graph with certain properties. The added parameters `INF` and `SUP` will be instantiated with number nodes, and specify lower and upper bounds for the nodes in the matched path. Parameters `NI` and `NS` specify the actual initial and final node numbers of the path, and are elsewhere required to satisfy $\texttt{INF} \leq \texttt{NI} \leq \texttt{NS} \leq \texttt{SUP}$. The predicate `sentence` is satisfied by every path in the input graph that begins at node `NI`, ends at node `NS`, and consists of three sub-paths matched by `subject`, `verb`, and `object`, in that order. If the match succeeds, the predicate `buildtree` defines `T` as a tree node, whose label `'sentence_1'` identifies the syntax rule, and whose subtrees are the parse trees `T1`,`T2`,`T3` of the constituent phrases.

Note that the translator moved the `verb` term—the starred item in (1)— to the beginning of the Prolog rule. (The proper order of the sentence's constituents is still ensured by the `NI`/`NS` arguments.) This feature was introduced to avoid infinite loops in syntax rules that begin with recursive optional terms — such as `subject`, which may be elided, and may be a subordinate clause beginning with a `subject`. Each recursive attempt to match a `subject` will have to go through the `verb` term, which cannot be elided and therefore will consume at least one word. Threfore the `subject`'s `NI` and `N1` arguments will be constrained to a strictly smaller range of indices at each recursive call.

6 Evaluation and Comparison

In order to evaluate Selva's performance on real-world texts, we created a small corpus of 80 inputs, by taking the first sentence from the second paragraph of newspaper and magazine articles. The entries averaged 14.7 words each (minimum 5, maximum 37); most of them used subordination, and many had non-trivial coordinations. These clauses were run through a preliminary version of our parser and grammar. In 28 cases, the parser failed to terminate, or did not find the correct parsing (the structure that a human parser would give, using all syntactic and semantic information available). For the remaining 52 cases where the correct structure was found, it produced 51.4 parsing per sentence, on the average (minimum 2, maximum 480).

Since this test was performed, we introduced the interval-constraint mechanism and re-wrote large parts of the grammar. Therefore, the above results should be viewed only as an indication that the basic premise – that it is feasible to generate all syntactically valid parsings – is quite realistic.

For comparison, we ran the same 80 clauses through the VISL parser. In almost all cases, the single derivation tree that was returned was at least syntactically possible. However, in about half of the cases, it did not match the correct tree, as it would be defined by a human parser — typically because of incorrect guesses about the nesting of prepositional phrases and subordinate clauses.

We also tried our corpus on the Curupira parser (version 1.0). That version seems configured to return only the first 4 parse trees found, in the order implied by the ordering of the rules in the grammar. According to the few tests which we have run so far, its coverage seems to be still incomplete. However the parser is still under development, and we expect its performance will improve considerably in future releases.

7 Future Work

We plan to continue to improve the grammar in light of systematic tests. In particular, we plan to tune it, by adding or excluding rules, so as to minimize the number of spurious derivation trees returned while improving the success rate. We also intend to use a more compact representation for the output, namely a single tree with `OR` nodes to encode the multiple choices allowed for each syntactic node. Such an encoding would allow us to generate and represent exponentially many parse trees for any n-word sentence, at polynomial cost.

Acknowledgments. We wish to thank Graça V. Nunes and the team at NILC — Núcleo Interdepartamental de Linguística Computacional of the University of São Paulo, in São Carlos — for kindly providing us a pre-release of the Curupira parser and allowing us to use the ReGra tagged dictionary. This work was supported in part by CAPES and CNPq.

References

1. E. Bick. *The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework.* PhD thesis, Aarhus Univ., 2000.
2. P. Cipro Neto. *Gramática da Língua Portuguesa.* Scipione, 1997.
3. W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer, 1994.
4. R. T. Martins, R. Hasegawa, and M. G. V. Nunes. CURUPIRA: Um parser funcional para a língua portuguesa. Technical Report NILC-TR-02-26, NILC-ICMC, Universidade Estadual de São Paulo, December 2002.
5. R.T. Martins, R. Hasegawa, M. G. V. Nunes, G. Montilha, and O. N. Oliveira Jr. Linguistic issues in the development of ReGra: A grammar checker for Brazilian Portuguese. *Natural Language Engineering*, 4(4):287–307, December 1998.
6. R. M. Mesquita. *Gramática da Língua Portuguesa.* Saraiva, São Paulo, 1999.
7. A. M. Perini. *Gramática Descritiva do Português.* Ática, 1996.
8. R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A Comprehensive Grammar of the English Language.* Longman, 1985.
9. L. A. Sacconi. *Nossa Gramática.* Atual Editora, São Paulo, 1984.
10. D. Santos. Um centro de recursos para o processamento computacional do português. *Datagrama Zero - Revista de Ciência da Informação*, 3(1), February 2002.