# INCREMENTAL PARTIAL PARSER OF UNRESTRICTED NATURAL LANGUAGE SENTENCES

Antonio Molina, Ferran Pla, Lidia Moreno, Natividad Prieto
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{amolina, fpla, lmoreno, nprieto}@dsic.upv.es

**Abstract**

One of the current focuses of research within natural language processing is the partial and robust parsing of sentences written in natural language. Partial parsing could be used in diverse applications as data extraction, machine translation, dialogue systems, etc. His main attractiveness is that is able to handle unrestricted sentences, that contain lexical errors or that present constructions not accepted by the defined grammar. Partial parsing is an alternative to the definition of wide coverage grammars whose definition is an expensive and complex task and that present well-known problems such as overgeneration, undergeneration and ambiguity. In this paper, we present a partial parser of unrestricted natural language sentences APOLN (**A**nalizador **P**arcial de **O**raciones en **L**enguaje **N**atural) which is based on finite-state machines. APOLN is an incremental parser that permits the compiling and inheritance of feature structures between levels of processing. We present the results of applying APOLN on an unrestricted Spanish corpus and we will use it in a speech dialogue system.

## 1 INTRODUCTION

S. Abney defines partial parsing as follows "*Partial parsing techniques aim to recover syntactic information efficiently and reliably from unrestricted text by sacrificing the completeness and depth of analysis".* A partial parser presents the following characteristics [2], [14]:

- It uses *robust parsing algorithms* which permit the analysis of unrestricted texts. This means that, independently from the structure of the sentence, the partial parser is able to get an interpretation, although it is a partial interpretation.
- These robust parsing algorithms are *more efficient* than global parsing algorithms.
- A partial parser works with *simpler grammars*, which are usually defined with regular patterns
- A parser has to use heuristics or procedures to combine the partial interpretations in order to build a final interpretation, that is, in order to connect non-adjacent parsed constituents. Achieving this could be necessary semantic information.
- A parser has to be completed with mechanisms that allow us to continue the analysis in spite of non-understandable segments of words. e.g. by identifying phrase boundaries.

While the output of a global parser is a complete analysis tree, if the sentence is syntactically correct, a partial parser postpones the attachment decisions between grammatical constituents if it does not have enough information. In this case, the output is a forest of subtrees which are not interleaved, that is, the trees do not share any nodes. Each tree represents a parsed fragment of the input. Segments of words that have not been recognized appear between the subtrees.

Partial parsing has been used in several **applications**: syntactic parsing of unrestricted texts [13], [6]; data extraction systems [20], [14]; machine translation [11]; acquisition of lexical information for solving attachment ambiguity [10], [20]; speech recongnition systems [19], [21], [4]; anaphora resolution [9]; text summarization, text categorization, etc.

## 2 PARTIAL PARSER OF UNRESTRICTED NATURAL LANGUAGE SENTENCES BASED ON FINITE-STATE MACHINES (APOLN)

APOLN is an incremental parser based on finite-state techniques that has been developed following the ideas described in [1], [3], [7], [8]. APOLN allows for syntactic parsing of unrestricted text. It is composed of several levels, a set of syntactic structures is recognized at each level and the output produced by a level *i* is the input to the level *i+1*. A level is defined by a set of patterns using regular expressions. **Figure 1** shows the APOLN scheme.

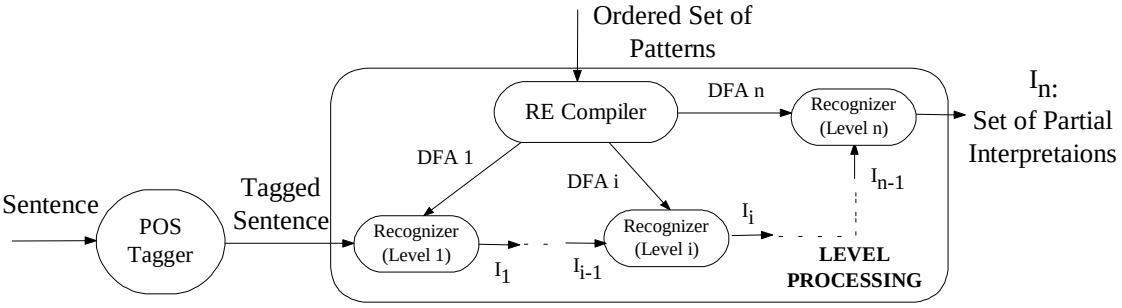


**Figure 1: APOLN scheme**

The first step is the lexical tagging performed by the *POS tagger* that will be described briefly in §2.1. The *Tagged Sentence* and an *Ordered Set of Patterns* which is divided into *n* levels form the input to the *Level Processing* module. The input to the first processing level is a *Tagged Sentence*, the input to a level *i* is the *Interpretation* produced by level *i-1* ($I_{i-1}$).

Each set of patterns of a specific level is compiled into a deterministic finite automaton (DFA). When the *Recognizer* module is executed for a level *i*, it takes $I_{i-1}$ and the $DFA_i$ as input. The output ($I_i$) represents the input in which the longest sequences of symbols that match a pattern (longest match, [1]) have been identified using boundary markers and syntactic tags. The final state reached determines the matched pattern.

### 2.1 POS Tagger description

A tagger can be considered as a translator that inputs strings from a certain language and outputs the corresponding sequence of lexical tags (grammatical categories). Generally, these categories are taken from a set defined previously by linguistic criteria. When a word can be assigned to different lexical categories, the disambiguation is solved by using the information of the context in which this word appears.

The tagging process involves two knowledge sources: the *language model*, which describes the possible (or probable) sequencing of the categories, and the *lexical model* which represents the relationships between the vocabulary of the application and the set of categories.

The language model is a stochastic regular grammar or finite-state automaton learnt automatically from data using grammatical inference techniques; in particular, we have used the ECGI algorithm, [16] [17] [18]. The model learnt generalizes the sequence of POS strings in the training corpus. In order to increase the coverage of the ECGI model, it has been extended and it has been smoothed by linear interpolation with a simple bigram model [15].

The lexical model has been estimated as usual from a supervised tagged corpus by computing words, categories and words per category frequencies.

Finally, the tagging process is carried out by Dynamic Programming Decoding, taking as input the output of the Morphological Analyser (MACO+) [5] that supplies for every word all the possible lexical tags.

### 2.2 Input Patterns Description

Patterns represent the syntactic constituents that should be identified from the input sentence. The symbols allowed for defining a pattern of level i are whatever lexical tags and whatever pattern which are defined at a previous level. In this way, patterns are *non-recursive* which allows for incremental parsing.

We have used the usual operators for the definition of the patterns: concatenation, Kleene closure (*), positive closure (+), union (|), one or more cases (?), and parentheses. The set of patterns defined is a set of regular definitions which are grouped by levels. Each level can be defined by several patterns. **Figure 2** shows the scheme for a certain level *i*, where $\mathbf{p_{i,j}}$ is a symbol that represents the *j* syntactic structure expressed at level *i*, and $\mathbf{r_{i,j}}$ is the regular expression that defines the pattern using the indicated operators and the non-recursivity constraint.

Level i //comment
$p_{i,1}$ -> $r_{i,1}$ //comment
$p_{i,2}$ -> $r_{i,2}$ //comment
...
$p_{i,n}$ -> $r_{i,n}$ //comment

**Figure 2: Level definition scheme**

Patterns can correspond to syntactic constituents such as noun phrases, adjective phrases, etc. or can be used to identify specific occurrences such as dates, entities, specific expressions, etc. which could be useful in data extraction systems.

Some of the patterns have been based on the modified concept of chunk [1]. In Spanish, a head or phrase could present non-recursive postmodifiers. Therefore, we redefine the concept of chunk as the core of "*a non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head, including post-head dependents that are not defined recursively using this constituent*".

### 2.3 Input and output string format description

Input and output formats is bracketed text, which is similar to the format used for parsing large corpora of text, e.g. Penn Treebank [12]. The input and the output of each level of processing is composed of a sequence of symbols $s_1\ s_2\ ...\ s_m$, where each $s_i$ can be a **lexical tag**, a **pattern** defined at a previous level, or a boundary mark (beginning, **[,** or ending, **],** marks). A pattern symbol always appears after an ending mark. So, if $s_1\ s_2\ ...\ s_m$, is an input string, $p_i$ is a pattern of level *i*, given that there exists a sequence of *k* symbols that matches $p_i$ from position *j*, the output would be the sequence $s_1\ s_2\ ...[\ s_j\ s_{j+1}...s_{j+k-1}\ ]p_i...\ s_m$. **Figure 3** represents a sentence which has been parsed after two levels of processing

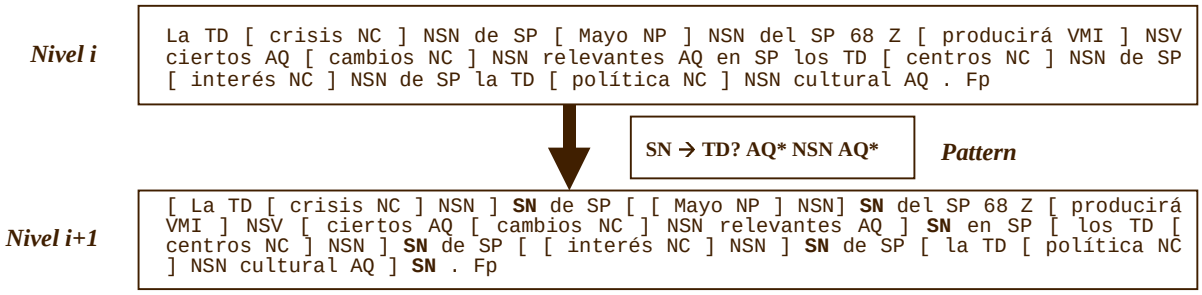


**Figure 3: Parsed sentence**

### 2.4 Compiling and Inheriting Feature Structures

The morphosyntactic features are necessary to parse sentences correctly. For instance, premodifiers and noun phrase head must agree in gender and number, the subject must agree with main verb, etc. Moreover, the use of morphosyntactic features can solve some parsing errors which are caused by the application of the longest match heuristic.

#### 2.4.1 Compiling Feature Structures

One possible solution would be to use lexical tags that contain morphosyntactic information.This supposes defining many patterns, one for each correct combination of features.

Our approach consists in including actions within the definition of the patterns, by means of a new operator, called *agreement operator*. This means that the transition between two states of the DFA is possible when the compatibility condition between two feature structures is true

(e.g. gender and number agreement). The current state stores the features associated to the last read symbol. The transition would be possible if the stored features were compatible with the features of the current symbol.

The *agreement operator*, noted by **&**, indicates the patterns in which the compatibility check should be done. The operator & is used in this way: **&p → r.** For instance, ***&****NSN → (NC | NP)*$^{+}$ this means that NC (*common noun*) and NP (*proper noun*) constituents must agree to form a NSN (*noun head*). We extend the DFA to include features and a compatibility check in transitions. The extended DFA is a 5-tuple **(Σ, Q, ($q_0$, $r_0$), F,δ),** where

Σ, is the alphabet (lexical tags and patterns symbol)

**Q**, is the set of states. Each state is the pair ($q_i$,$r_i$), where $q_i$ identifies the state and $r_i$ is the associated Feature Structure.

**($q_0$, $r_0$)** ∈ Q, is the initial state containing an initial Feature Structure $r_0$.

**F**⊂Q, is the set of final states and δ is the transition function that is defined as:

δ(($q_i$,$r_i$),s) = ($q_j$, rme($r_i$,rasgo(s))) if compatible($r_i$, rasgo(s)): ($q_i$, $r_i$), ($q_j$ , $r_j$) ∈ Q, s ∈ Σ,

**rasgo(s)** returns the feature structure associated to symbol s.

**rme($r_i$, $r_j$)** returns the most specific or restricting feature between $r_i$ and $r_j$

**compatible($r_i$,$r_j$)** check the compatibility between the features $r_i$ and $r_j$.

The DFA compiled from *&NSN → (NC | NP)+* is *({NC,NP}, {(q0, r0), (q1, r1)}, (q0, r0), {(q1, r1)},δ)* where δ is:

δ((q0,r0),NP) = (q1, rme(r0,rasgo(NP ))) if compatible(r0,rasgo(NP) )

δ((q0,r0),NC) = (q1, rme(r0,rasgo(NC ))) if compatible(r0,rasgo(NC) )

δ((q1,r1),NP) = (q1, rme(r1,rasgo(NP ))) if compatible(r1,rasgo(NP) )

δ((q1,r1),NP) = (q1, rme(r1,rasgo(NC ))) if compatible(r1,rasgo(NC) )

This technique allows us to check morphosyntactic agreement using the feature information contained in lexical tags. Moreover, it could be extended to others such as semantic features to verify semantic compatibility between constituents.

### 2.4.2 Feature Inheritance

Patterns must inherit the features associated to their constituents. These features are necessary at higher levels where the patterns become constituents of other patterns. In general, a pattern inherits the features of its head, e.g. the pattern *NSN → (NP | NC)+* will inherit the features of NC or NP. The *inheritance operator*, noted by **$**, indicates the symbol which the features are inherited from, e.g. *NSN → ($NP | $NC)+*.

## 3 RESULTS OF APPLYING *APOLN* TO UNRESTRICTED TEXT

The Spanish corpora used in this work were LEXESP and CPirápides [6]. A subset of LEXESP has been used to learn the language model and the lexical model of the tagger. It consists of 75 Kw manually tagged words. The tagset used is composed of 62 PAROLE tags [12]. CPirápides has been used to test both the tagger and the partial parser. This is a very simple corpus consisting of 5 Kw and an overall lexical ambiguity of 1.58 tags/word. We have used this one because the tagging and the parsing have been manually supervised. So we can compare easily the performance of our system. The syntactic structures identified in CPirápides were Noun Phrase (SN), Verbal Heads (NSV), Prepositional Phrase (SPR), Adjective Phrase (SADJ), Infinitive Heads (NSVI), Adverbial Phrase (SADV), Conjunctions and Relative Pronouns (SUB). We have designed a set of three levels of patterns in order to recognize these structures, (see **Figure 4**).

*Level 1*

**NSV** -> PP?PP?(((VMI|VMS|VMC|VMM)(((VMN|VAN)(VMG|VMP)?)|
VMG)?)|((VAI|VAS|VAC|VAM)(VMP|VAP|VMG|VAG|VMN|VAN)*)|
(VMI|VMS|VMC|VMM|VAI|VAS|VAC|VAM)CS(VMN|VAN))

**NSVI** -> (VMN|VAN)

**SUB** -> (SP? CS)|(SP? TD? PR)

**SN** -> ((DD|DP|DT|DE|DI|D0|TD|TI|MC|MO)* ((RG? (AQ | VMP)+)? | Z? |
RG) (W|(NP|(NC (CC NC)*))+) (RG? (AQ | VMP)+)?)|(PP|PD|PX|PI|PT|P0)

*Level 2*

**SPR** -> SP (SN | NSVI)

**SADJ** -> RG? (AQ | VMP)

*Level 3*

**SADV** -> (SP RG) | (RG RG?)

**Figure 4: Level Definition**

In order to test the capabilities of the proposed approach we carried out the following experiments. In the Experiment 1, we have considered as input a text without tagging errors (Supervised Tagging, ST) in order to evaluate only the performance of the syntactic parser. In the Experiment 2, we have used APOLN taking as input the output of our tagger (Unsupervised Tagging, UT). **Table 3-1** summarizes the precision and the recall rates for each syntactic structure studied in both experiments. We can see how the errors of the tagger (about 1.6%) contribute to decrease the performance of the parser. The high error rates achieved for SADJ, SPR and SN are because of the fact that the tagger confuses a common noun with an adjective (70% of tagging errors are of this type). Moreover, the low precision achieved for SADV is

because negation has been interpreted as an adverbial phrase and CPirápides considers it as a constituent of the verbal phrase. On the other hand, the speed of analysis depends on the number of levels. Using the three levels defined in **Figure 4**, the parser achieves about 4700 words/second on a Pentium 120 Mz.

| **Corpus CPirápides** | | **NSV** | **NSVI** | **SN** | **SUB** | **SPR** | **SADJ** | **SADV** |
|---|---|---|---|---|---|---|---|---|
| ***Experiment 1*** | **Precision (%)** | 98,6 | 100.0 | 99,9 | 100.0 | 100.0 | 100.0 | 62,5 |
| **ST + APOLN** | **Recall (%)** | 98,3 | 100.0 | 99,0 | 94,7 | 98,7 | 66,7 | 95,2 |
| ***Experiment 2*** | **Precision (%)** | 98,3 | 100.0 | 97,7 | 94,1 | 99,8 | 4,4 | 51,6 |
| **UT + APOLN** | **Recall (%)** | 96,7 | 100.0 | 96,3 | 84,2 | 92,7 | 66,7 | 76,2 |

**Table 3-1: Precision and recall**

## 4 CONCLUSIONS AND FUTURE WORK

In this paper we have presented an incremental parser, APOLN, which is based on finite-state machines. APOLN is able to identify syntactic structures in unrestricted text. These structures can be defined by several levels of processing which gives flexibility to the parser: the user could define their own levels easily. Our approach includes feature compilation and inheritance. This characteristic has been proven with morphosyntactic features but could be extended to any kind of feature which is linked to the words of the input, for example, semantic information.

The experiments performed on corpus CPirápides have given good results identifying phrases. Also, in order to assess more conclusively the capabilities of the proposed approach, we are working on a more complex corpus, LEXESP, and the preliminary results obtained are promising. On the other hand, we think that this parser could be useful in a dialogue speech system that works with unrestricted and ill-formed sentences. In this sense, we are working on a semantically restricted task to extract the meaning of the sentences by filling in of case-frames.

Moreover, we are developing a parsing system that allows us to completely parse an unrestricted corpus. The system use APOLN as first step of processing. The entire parsed corpus could be useful as an information source for treating linguistic phenomena and for developing inductive methods based on corpus.

## 5 ACKNOWLEDGEMENTS

This paper has been supported by the Spanish CICYT project TIC97-0671-C02-01/02. We would like to thank Horacio Rodríguez, Irene Castellón and Lluís Padró for his valuable advices and for allowing us to use LEXESP, CPirápides and the toolkit MACO+.

# 6 REFERENCES


[1] Abney, S. "Partial Parsing via Finite-State Cascades". In *ESSLLI'96 Robust Parsing Workshop.* 1996.

[2] Abney, S. "Tagging and Partial Parsing". *Corpus-Based Methods in Language and Speech processing*. S. Young y G.Bloothooft Eds. Kluwer Academic Publishers 1997.

[3] Aït-Mokhtar, S. Chanod, J.P. "Incremental Finite-State Parsing". In *Proc. of the Fifth Conference on Applied Natural Language Processing.* Washington D.C., USA, 1997.

[4] Baggia, P., Rullent, C. "Partial Parsing as a Robust Parsing Strategy". *Proc. of ICASSP'93*.

[5] Carmona J., Cervell S., Màrquez, L., Martí, M.A., Padró, L., Placer, R., Rodríguez, H., Taulé, M., Turmo, J. "An Environment for Morphosyntactic Processing of Unrestricted Spanish Text". In *LREC'98*, 1998.

[6] Castellón, I. Civit, M. Atserias, J. "Syntactic Parsing Of Unrestricted Spanish Text". In *LREC'98*, 1998.

[7] Chanod, J.P., Tapanainen, P. "A Robust Finite-State Parser for French". In *ESSLLI'96 Robust Parsing Workshop*. 1996.

[8] Ejerhed, E.I. "Finding Clauses in Unrestricted Text by Finitary and Stochastic Methods". In *Proc. of Second Conference on Applied Natural Language Processing.* ACL, 1988.

[9] Ferrández, A., Palomar, M., Moreno, L. "Anaphor Resolution In Unrestricted Texts With Partial Parsing". *Proc. of the 36$^{TH}$ Annual Meeting of the Association for Computational Linguistics.* Montreal, 1998

[10] Hindle, D., Rooth, M. "Structural Ambiguity and lexical relations". *Computational Linguistics nº 18*, 1993.

[11] Light, M., "CHUMP: Partial Parsing and Underspecified Representations," In *Proc. of the ECAI-96 Workshop: Corpus-Oriented Semantic Analysis*, 1996.

[12] Martí M.A.,Rodríguez H.,Serrano J. “Declaración de categorías morfosintácticas”. Doc.ITEM nº2.UPC, UB.

[13] Marcus, M.P., Santorini, B. Marcinkiewicz, M.A.. "Building a Large Annotated Corpus of English: The Penn Treebank", *Computational Linguistics* nº 19, 1993.

[14] McDonald, D. "An efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts". *Third Conference on Applied Natural Language Processing*, 1992.

[15] Pla, F. Prieto, N. "Using Grammatical Inference Methods for Automatic Part-of-Speech Tagging". In *LREC'98*, 1998.

[16] Prieto N. & Vidal E. "Learning Language Models through the ECGI Method". *Speech Communic.*, 11, 1992.

[17] Rulot H., Prieto N. & Vidal E. "Learning accurate finite-state structural models of words through the ECGI algorithms". *Proc. of International Conference on Acoustic and Speech Signal Processing*, 1989.

[18] Rulot H., Vidal E. "Modelling (sub)string-length-based constraints through a Grammatical Inference Method". In *Pattern Recognition: Theory and Applications*. Eds. Devijver & Kittler, Springer Verlag, 1987.

[19] Seneff S. "A Relaxation Method for Understanding Spontaneous Speech Utterances". En *Proc. of Speech and Natural Language Workshop.* N.Y. 1992.

[20] Weischedel, R., Ayuso, D., Bobrow, R., Boisen, S., Ingria, R., Palmucci, J. "Partial Parsing: A Report on Work in Progress". In *Proc. DARPA Speech and Natural Language Workshop*, 1991.

[21] Weischedel, R., Ayuso, D., Boisen, S., Fox, H., Ingria, R.,. "A New Approach to Text Understanding". In *Proc. of Speech and Natural Language Workshop.* N.Y. 1992.