APOLN: A Partial Parser Of Unrestricted Text

Antonio Molina, Ferran Pla, Lidia Moreno, Natividad Prieto
Department of Information Systems and Computation
Valencia University of Technology
{amolina, fpla, lmoreno, nprieto}@dsic.upv.es

Abstract

In this paper, we present APOLN (**A**nalizador **P**arcial de **O**raciones en **L**enguaje **N**atural): a partial parser of unrestricted natural language sentences based on finite-state techniques. Partial parsing has been used in several applications: syntactic parsing of unrestricted texts, data extraction systems, machine translation, solving the attachment ambiguity, speech recognition systems, text summarization, etc. The main attractiveness of partial parsing 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. We present APOLN as a tool that can be used to construct syntactically annotated corpora from lexically tagged corpora. We also present the results (precision and recall rates) of applying APOLN on unrestricted Spanish corpora, and how tagging errors influence the performance of the parser.

1 Introduction

One of the current focuses of research within natural language processing is the partial and robust parsing of sentences in natural language. Partial parsing is a technique that *aim to recover syntactic information efficiently and reliably from unrestricted text by sacrificing the completeness and depth of analysis* [Abney97]. Partial parsing uses more robust and more efficient algorithms than global parsing. It works with simpler grammars usually defined with regular patterns. Moreover, it handles mechanisms that allow us to continue the analysis in spite of non-understandable segments of words.

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.

One of the applications of partial parsing is the syntactic parsing of unrestricted corpora. Partial parsing can be used as a first step to construct a syntactically parsed corpus (with complete analysis trees). An overview of the main applications of partial parsing can be seen in [Molina98].

2 System Description

APOLN allows for syntactic parsing of unrestricted text. APOLN is an incremental parser based on finite-state techniques (see [Abney96], [Aït-Mokhtar97], [Chanod96], [Ejerhed88]). APOLN processes a set of levels. At each level a set of syntactic structures is recognized and the output produced by a level *i* is the input to the next level *i+1*. A level is defined by a set of patterns using regular expressions.

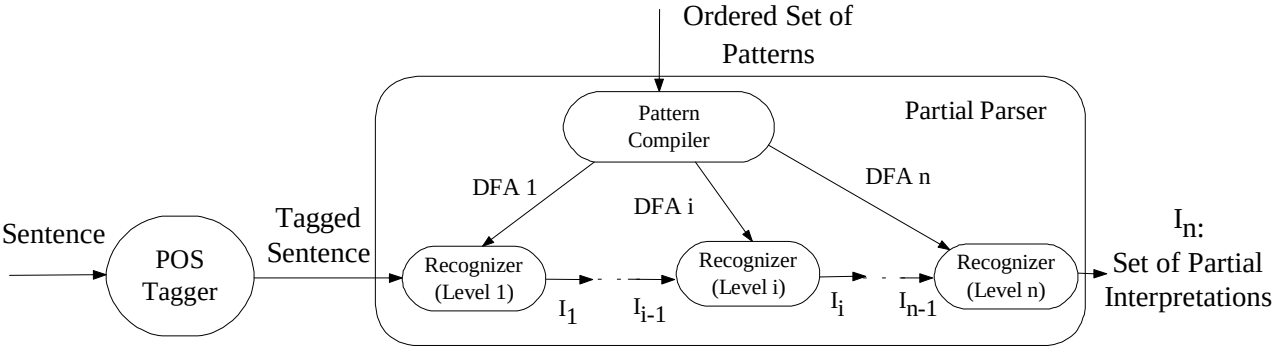

Figure 1: APOLN scheme

Figure 1 shows an overview of the system. The first step is the lexical tagging performed by the *POS tagger* described in **Figure 2**. The *Tagged Sentence* and an *Ordered Set of Patterns* which is divided into *n* levels forms the input to the *Level Processing* module. The input to the first level is a *Tagged Sentence*. The input to a level *i* is the *Interpretation* produced by level *i-1* (I_{i-1}). The output of the system is a *Set of Partial Interpretations* that represents the syntactic parse of the *Sentence* using bracketed format.

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, [Abney96]) have been identified using boundary markers and syntactic tags. The final state reached determines the matched pattern.

The morphosyntactic features are necessary to parse sentences correctly. Also, they help to solve some parsing errors which are caused by the application of the longest match heuristic. Therefore, our approach includes actions within the definition of the patterns, by means of *agreement operator* and *inheritance operator*. The agreement operator means that the transition between two states of the DFA is possible when the compatibility condition between two feature structures is true. The inheritance operator indicates the features that have to be inherited from one level to higher levels.

2.1 POS Tagger description

A tagger can be considered as a translator that inputs strings from a certain language (Unrestricted Text) and outputs the corresponding sequence of lexical tags or grammatical categories (Text tagged). 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. **Figure 2** shows an scheme of the tagger used.

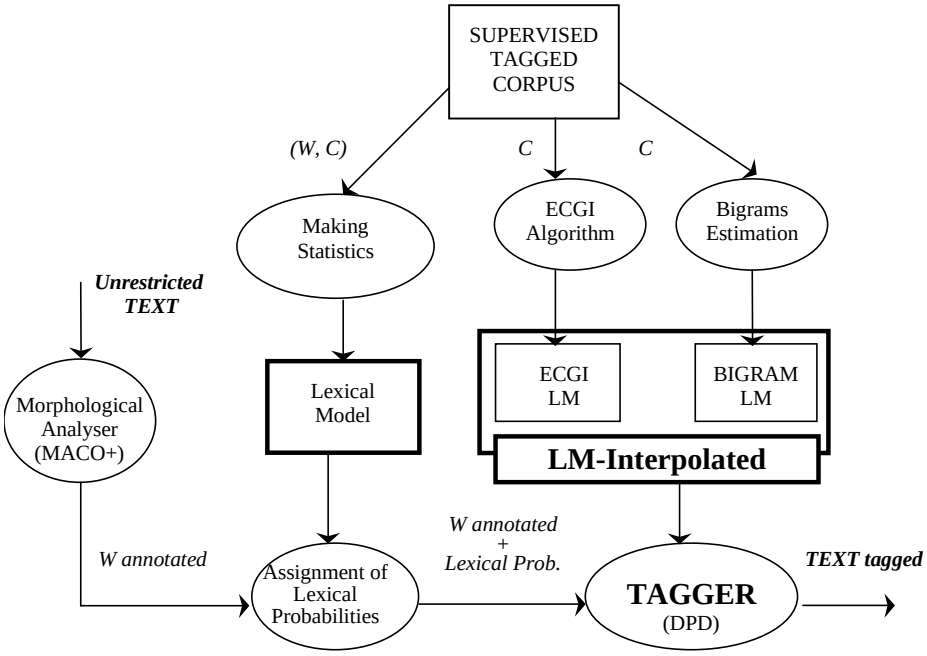

Figure 2: POS tagger description

The tagging process involves two knowledge sources: the *language model (LM)*, 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 used is a stochastic regular grammar or finite-state automaton learnt automatically from data (sequences of categories C) using grammatical inference techniques; in particular, we have used the ECGI algorithm, [Roulot89], [Prieto92]. 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 smoothed by linear interpolation with a simple bigram model. The interpolation factor has been estimated experimentally. Then, a renormalization process is applied in order to maintain the stochastic consistency. A more detailed description can be seen in [Pla98].

The lexical model has been estimated as usual from a manually tagged corpus (W,C) by computing words, categories and words per category frequencies.

MACO+ [Carmona98], is a modular morphological analyser that initially perform a proper segmentation of the input text into tokens. It identifies punctuation marks, lexical units, dates, abbreviations, numbers, proper nouns, etc. Also, it supplies all the possible lexical tags for every token detected. Then, the lexical probability of each output token given by MACO+ is calculated taking into account the Lexical Model.

Finally, the tagging process is carried out by Dynamic Programming Decoding (DPD), taking as input the output tokens of MACO+ with their respective lexical probabilities.

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 3** 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 3: 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.

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 [Marcus93]. 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 $s_1\ s_2\ ...[\ s_j\ s_{j+1}...s_{j+k-1}\ ]p_i...\ s_m$. **Figura 4** represents a sentence which has been parsed after two levels of processing

Level i

```
La TD [ crisis NC ] NSN de SP [ Mayo NP ] NSN del SP 68 Z [ producirá VMI ] NSV
ciertos AQ [ cambios NC ] NSN relevantes AQ en SP los TD [ centros NC ] NSN de SP
[ interés NC ] NSN de SP la TD [ política NC ] NSN cultural AQ Fp
```

SN → TD? AQ* NSN AQ* ***Pattern***

Level i+1

```
[ La TD [ crisis NC ] NSN ]SN de SP [ [ Mayo NP ] NSN]SN del SP 68 Z [ producirá
VMI ] NSV [ ciertos AQ [ cambios NC ] NSN relevantes AQ ] SN en SP [ los TD [
centros NC ] NSN ]SN de SP [ [ interés NC ] NSN ]SN de SP [ la TD [ política NC
] NSN cultural AQ ]SN . Fp
```

Figure 4: 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) $\in$ Q, is the initial state containing an initial Feature Structure r_0.

F$\subset$Q, is the set of final states and δ is the transition function that is defined as:

$\delta((q_i,r_i),s) = (q_j, rme(r_i,rasgo(s)))$ if compatible(r_i, rasgo(s)): (q_i, r_i), (q_j , r_j) $\in$ Q, s $\in$ 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:

$\delta((q0,r0),NP) = (q1, rme(r0,rasgo(NP))) if compatible(r0,rasgo(NP))$

$\delta((q0,r0),NC) = (q1, rme(r0,rasgo(NC))) if compatible(r0,rasgo(NC))$

$\delta((q1,r1),NP) = (q1, rme(r1,rasgo(NP))) if compatible(r1,rasgo(NP))$

$\delta((q1,r1),NP) = (q1, rme(r1,rasgo(NC))) if compatible(r1,rasgo(NC))$

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)+*.

4 Experimental Results

In order to test our system, we have used the Spanish corpora CPirápides and LEXESP. CPirápides is a simple corpus consisting of 5 Kw. On the contrary, LEXESP is a multidisciplinary corpus, which contains 5.5 Mw of written material, including general news, sports news, scientific articles, etc. LEXESP presents more complex sentences than CPirápides.

The tagset used in both corpora is composed of 62 PAROLE tags [Martí98]. The syntactic structures identified were Noun Phrase (SN), Verbal Heads (NSV), Prepositional Phrase (SPR), Adjective Phrase (SADJ), Infinitive Heads (NSVI), Adverbial Phrase (SADV), Conjunctions and Relative Pronouns (SUB).

In this work we have redefined the set of non-recursive patterns used in the preliminary experiment presented in [Molina99] in order to increase the coverage of the parser. It is showed in Figure 5.

Level 1
NSV -> (VMI|VMS|VMC|VMM|VAI|VAS|VAC|VAM)CS(VMN|VAN)(VMG|VMP)?
NSV -> PP?PP?(((VMI|VMS|VMC|VMM)(((VMN|VAN)(VMG|VAG|VMP)?)|(VMG | VAG))?)|((VAI|VAS|VAC|VAM)(VMP|VAP|VMN|VAN)*(VMG | VAG)?))
NSV -> (VMG|VAG)
NSVI -> (VMN|VAN)(((CC|Fc) (VMN|VAN))* CC (VMN | VAN))?
SADJ -> RG? (AQ | VMP) (((CC|Fc) RG? (AQ | VMP))* CC RG? (AQ | VMP))?
NSN -> ((NP (((CC|Fc) NP)* (CC NP))?) | (NC (((CC|Fc) NC)* (CC NC))?))+
Level 2
SN -> TD (PX|PI) (AQ|VMP)?
SN -> DI (PP|PI|PD|P0)
SN -> (DD|DP|DT|DE|DI|D0|TD|TI|(MC* (CC MC)?)|MO)* (SADJ? | Z? | RG) NSN SADJ?
SN -> (PP|PD|PX|PI|PT|P0)|((DD|DP|DT|DE|DI|D0|TD|TI) P0)
SN -> (DD|DP|DT|DE|DI|D0|TD|TI|MO)* (Z| W | MC* (CC MC)?)
SN -> TD SADJ
Level 3
SPR -> SP TD? (PP|PI) D0
SPR -> SP SN
SPR -> SP SADJ
SPR -> SP NSVI
Level 4
SUB -> (SP? CS)|(SP? TD? PR)
Level 5
SADV -> (SP RG) | (RG RG?)

Figure 5: Levels of patterns

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. CPirápides and a subset of LEXESP (3 Kw) has been used to test our system.

We carried out two kinds of experiments. In the first one we have considered as input a text without tagging errors (Manually Corrected Tagging, CT) in order to evaluate only the performance of the syntactic parser. In the second one, we have used APOLN taking as input the output of our

tagger (Tagger Output, TO). In **Table 1** and **Table 2**, we summarize precision[1] an recall[2] rates per pattern obtained in the experiments defined above.

We have revised the errors produced by the parser in both experiments. In CT experiments (Experiment 1 and Experiment 3) the most common errors are due to identify incorrectly adverbial locutions and some adjective phrases. Also, some compound nouns and compound adjective phrases are incorrectly attached. We can not solve this kind of ambiguity with the information provided by the lexical tags, because it would be necessary other information sources (e.g. semantic or contextual information). Moreover, in TO experiments (Experiment 2 and Experiment 4) the performance of the parser decreases because of the tagging error (0.8% in CPirápides and 3.0% in LEXESP). The most usual errors of the tagger are to confuse: adjectives and nouns, adjectives and verbs and adjectives and adverbs. Mainly, this affects to the precision and recall of SADJ.

Corpus CPirápides		**NSV**	**NSVI**	**SN**	**SUB**	**SPR**	**SADJ**	**SADV**
Experiment 1	**Precision (%)**	99.6	100.0	99.0	100.0	99.0	100.0	95.2
CT + APOLN	**Recall (%)**	97.7	100.0	98.5	94.7	98.7	66.7	95.2
Experiment 2	**Precision (%)**	99.6	100.0	98.5	94.1	98.3	66.7	94.7
TO + APOLN	**Recall (%)**	97.1	100.0	97.6	84.2	95.5	66,7	85.7

Table 1: Precision and recall for CPirápides

Corpus LEXESP		**NSV**	**NSVI**	**SN**	**SUB**	**SPR**	**SADJ**	**SADV**
Experiment 3	**Precision (%)**	97.6	100.0	98.9	100.0	96.3	77.6	100.0
CT + APOLN	**Recall (%)**	97.6	100.0	97.5	100.0	95.8	80.9	97.6
Experiment 4	**Precision (%)**	94.3	85.7	92.0	99.4	92.9	64.2	95.0
TO + APOLN	**Recall (%)**	94.7	100.0	89.4	100.0	92.4	72.3	92.7

Table 2: Precision and recall for LEXESP

4 Conclusions and Future Work

In this paper we have presented an incremental partial parser based on finite-state machines, that is able to identify syntactic structures on unrestricted text. The experiments performed have given good results identifying phrases, although the amount of available test set (supervised and syntactically parsed text) could be scarce to provide statistically significant results.

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.

[1] Precision: total number of correct tags given by the parser / total number of tags given by the parser * 100

[2] Recall: total number of correct tags given by the parser / total number of tags in reference corpus * 100

5 Acknowledgements

This paper has been supported by the Spanish CICYT project TIC97-0671-C02-01/02

References

[Abney96] Abney, S. "Partial Parsing via Finite-State Cascades". In *ESSLLI'96 Robust Parsing Workshop.* 1996.

[Abney97] Abney, S. "Tagging and Partial Parsing". *Corpus-Based Methods in Language and Speech processing*. S. Young y G.Bloothooft Eds. Kluwer Academic Publishers 1997.

[Aït-Mokhtar97] 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.

[Carmona98] 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.

[Chanod96] Chanod, J.P., Tapanainen, P. "A Robust Finite-State Parser for French". In *ESSLLI'96 Robust Parsing Workshop*. 1996.

[Ejerhed88] 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.

[Marcus93] Marcus, M.P., Santorini, B. Marcinkiewicz, M.A.. "Building a Large Annotated Corpus of English: The Penn Treebank", *Computational Linguistics* nº 19, 1993

[Martí98] Martí M.A.,Rodríguez H.,Serrano J. “Declaración de categorías morfosintácticas”. Doc.ITEM nº2.UPC, UB, 1998.

[Molina98] Molina, A. Moreno, L. “Técnicas de Análisis Parcial en Procesamiento del Lenguaje Natural”. Internal Report DSIC-II/30/98. Septiembre 1998.

[Molina99] Molina, A., Pla, F., Moreno L., Prieto N., “Incremental partial parser of unrestricted natural language sentences”. *In SNRFAI99*, Bilbao 1999.

[Pla98] Pla, F. Prieto, N. "Using Grammatical Inference Methods for Automatic Part-of-Speech Tagging". In *LREC'98*, 1998.

[Prieto92] Prieto N. & Vidal E. "Learning Language Models through the ECGI Method". *Speech Communic.*, 11, 1992.

[Rulot89] 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.

Appendix: Examples of usual parsing and tagging errors.

Wrong Tag: NC →AQ

[Esta DD envidia NC] SN [de SP la TD que PR] SUB [todos PI] SN [somos VAI] NSV **[víctimas NC] SN y CC [verdugos AQ] SADJ** [se PP ha VAI cebado VMP] NSV [en SP tí PP] SPR [singularmente RG] SADV , Fc ...

... [víctimas NC y CC verdugos NC] SN ...

Wrong Tag: RG →AQ

[No RG] SADV [estoy VMI] NSV **[seguro RG] SADV** [de SP que PR] SUB [las TD generaciones NC] SN [de SP hoy RG] SADV [sepan VMS] NSV [del SP poderoso AQ influjo NC] SPR ...

... [seguro AQ] SADJ ...

Wrong Tag: VMS → AQ

Y CC [entre SP sus DP disfraces NC] SPR [no RG] SADV [es VAI] NSV el TD **[menos RG] SADV [frecuente VMS] NSV** el TD [de SP la TD ideología NC] SPR . Fp

... [el TD menos RG frecuente AQ] SN ...

Wrong Tag: AQ → NC

[Un TI cuarto NC] SN y CC [último AQ punto NC] SN [me PP parece VMI] NSV [interesante AQ] SADJ [destacar VMN] NSVI [a SP este DD respecto NC] SPR (Fap [como CS] SUB [lo PP he VAI hecho VMP] NSV [en SP mi DP mencionado VMP libro NC El_Continente NP vacío AQ] SPR) Fcp . Fp

[Un TI cuarto AQ y CC último AQ punto NC] SN ...

Syntactic Error: Compound Noun

[Para SP evitarlo VMN] SPR , Fc [conspicuos AQ fascistas NC] SN , Fc [monárquicos AQ] SADJ [a SP la TD violeta NC] SPR , Fc [marxistas AQ] SADJ **[de SP provincias NC y CC católicos NC wojtilianos AQ]** SPR [se PP han VAI venido VMP confabulando VMG] NSV [en_eso RG] SADV [que CS] SUB [ahora RG] SADV [llaman VMI] NSV [pacto NC] SN [a SP la TD griega NC] SPR ...

... [de SP provincias NC] SPR y CC [católicos NC wojtilianos AQ] SN ...

Syntactic Error: Compund Adjective

[No RG] SADV [estoy VMI] NSV [seguro RG] SADV [de SP que PR] SUB [las TD generaciones NC] SN [de SP hoy RG] SADV [sepan VMS] NSV **[del SP poderoso AQ influjo NC] SPR , Fc [intelectual AQ y CC moral AQ] SADJ** , Fc [que CS] SUB [tu DP magisterio NC universitario AQ] SN [ejerció VMI] NSV [sobre SP la TD disidencia NC juvenil AQ] SPR ...

... [del SP poderoso AQ influjo NC , Fc intelectual AQ y CC moral AQ] SPR ...

Syntactic Error: Adverbial Locution

Y CC [es VAI] NSV [sobre SP esto PD] SPR [sobre SP lo PP] SPR [que PR] SUB , Fc [resuelto VMI] NSV [el TD breve AQ paréntesis NC] SN [de SP mis DP cartas NC] SPR , Fc [me PP gustaría VMC ahondar VMN] NSV **un TI [poco RG más RG] SADV** . Fp

... [Un_poco_más RG] SADV ...