Automatic Definition Extraction and Crossword Generation From Spanish News Text

Jennifer Esteche, Romina Romero, Luis Chiruzzo, Aiala Rosá
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
{jennifer.esteche, rromero, luischir, aialar}@fing.edu.uy

Abstract

This paper describes the design and implementation of a system that takes Spanish texts and generates crosswords (board and definitions) in a fully automatic way using definitions extracted from those texts. Our solution divides the problem in two parts: a definition extraction module that applies pattern matching implemented in Python, and a crossword generation module that uses a greedy strategy implemented in Prolog. The system achieves 73% precision and builds crosswords similar to those built by humans.

Keywords: Definition Extraction, Crossword Generation, Natural Language Processing.

1 Introduction

Crossword puzzles are, besides a pastime, a didactic tool that could be used to teach concrete topics, in the case of thematic crosswords, or to acquire vocabulary and language proficiency, in the case of general crosswords. On the other hand, definition extraction facilitates the access to information of interest, which is why it is an interesting topic by itself.

In order to generate crosswords in a totally automatic way it is necessary to solve two very different tasks: first extract the definitions from natural language texts, then generate the grid using the extracted definitions.

The task of definition extraction is part of the more general area of Information Extraction, and consists in recognizing a set of <definition, definendum> pairs inside a text, where the *definendum* is a term that appears in the text and its *definition* is a fragment of text that describes it [1]. In order to automatically extract these <definition, definendum> pairs we must identify the relation between the text fragments and determine which elements can be abstracted in order to characterize this relation, working at different levels of text analysis. For example, a definendum could be the term *"Hollande"* and its definition the fragment *"presidente de Francia" / "president of France"*. This pair could be extracted from the following sentence: *El presidente de Francia, François Hollande, habló en conferencia de prensa. / The president of France, François Hollande, gave a press conference.*

On the other hand, the crossword puzzle generation is basically an algorithmic problem: the aim is to find an efficient way of selecting a set of words from a list and distribute them inside a grid subject to certain restrictions. These restrictions imply some basic crossword characteristics, e.g. no invalid words should be formed, and other aesthetic characteristics, e.g. there should not be too many black squares. It is possible to see a way of modeling these characteristics as a constraint satisfaction problem ([2], [3]).

This work describes a system that allows to extract <definition, definendum> pairs from news text in Spanish and use those definitions to generate a crossword puzzle in a totally automated way. These crosswords use as many of the extracted definitions as possible, while complementing with external definitions when necessary. In order to do this we employ an auxiliary list of words, mainly short words that help fill blanks in the puzzle, since the number of extracted definitions might not be enough to cover the extent of the whole grid. Notice that the extracted definitions are tightly related to the news text they come from, so that sometimes they might not make sense to someone who has not read the text.

As far as we know, this is the only work that addresses the complete process of crosswords building (extracting definitions and creating the crossword) for the Spanish language.

The rest of the paper is organized in the following way: Section 2 shows a brief summary of the state of the art in the definition extraction and crossword generation tasks. Section 3 provides a description of the

whole system. Section 4 shows the process of definition extraction. Section 5 shows the crossword generation. Section 6 presents some conclusions and future work. Appendix A presents a full description of the patterns.

2 Related Work

2.1 Definition Extraction

Definition extraction is an active research area. Much of the research has been carried for the English language, although some authors have worked for other languages as well. Concerning the used techniques, many works in the area apply pattern matching. In some cases there is a first set of patterns which are defined manually, and then a second stage when machine learning methods are used to find other definitions or new patterns ([4], [1]). Other authors base their work in a predefined list of patterns [5][6]. Our work is essentially based in pattern matching. Although we explored the possibility of applying a later process of automatic pattern detection, this process did not give good results.

Some researchers use annotated corpora that include documents from different domains ([7], [5], [8]). Others base their work on text that has certain predefined structure, for example: using dictionaries, thesaurus or encyclopedic text whose structure is clear ([5], [9]). When using domain specific text ([10], [11], [12], [6]) there are certain linguistic patterns that facilitate the extraction of definitions. Many authors that use pattern matching use regular expressions due to their simplicity and efficiency ([5], [11], [13]). In our work, we used recursive regular expressions, which are more expressive than classical regular expressions.

Many authors agree that working with a language different than English is more challenging because of the lack of available tools and resources for those languages, which entails an additional difficulty ([5], [4], [13]).

Definition extraction methods usually depend strongly on the target language and its grammar. In particular, using morphosyntactic (*POS-tagger*) and syntactic (*parser*) analyzers, and then applying morphosyntactic patterns, seem to be the most commonly used techniques for this task. However, the authors of [4] describe a mechanism that tries to use as little syntactic information as possible, depending heavily on lexical information instead. This forced them to apply data mining techniques to compensate for the poor generalization achieved using this method, and to consider at least number and gender features in the patterns. As well as this, the work of [1] -focused on German law text-, besides using extraction rules, describes an ontology based approach, exploiting the relations between concepts. Furthermore, they use a parser that returns semantic information. This approach achieved an average of 46% precision and, when only the most effective rules are used, the precision increases to more than 70%. Focusing on a particular domain like [6], which extracts definitions from consumer-oriented medical articles, could also lead to better results. They report a precision of 87%, which might be in part attributed to the nature of the corpus.

On the other hand, [14] and [12] propose using dependency parsing and consider the relative order of words in order to determine which word is being defined and what are the boundaries of the definition. The authors of [12] describe a layered system, the last of this layers assigns previously identified *chunks*[1] their syntactic function (subject, nominal predicate, verbal predicate, etc.). They report a precision of 53% using this method. Our approach includes some linguistic tools as well: a clause segmentation and a constituent parser.

The work of [13] proposes, however, to manually develop a partial grammar that could be applied together with a set of classifiers that will vote which classification is the correct one (whether a given sentence is a definition or not). This is an interesting approach as it tries to leverage both manual and statistical techniques, but their tests were performed over a relatively small corpus, where the machine learning approach cannot work at its best, obtaining 19.94% precision and 69.23% recall.

Many authors perform a classification of definitions, separating the definitions induced by the verb *"to be"* as a connector, the ones induced by verbs different than *"to be"* and the ones that use punctuation marks to separate the defined term and its definition. This categorization is used in [7] and [11], the former is based on Portuguese text while the latter uses English text. The work of [7] proposes developing three distinct regular grammars, one for each of the three types of definitions. This approach achieved a precision of 14% and a recall of 86%.

Notably, the approach in [11] does not use pattern matching but evolutionary algorithms. The algorithm is used to train a classifier that distinguishes between sentences that contain a definition and others that do not. Results are very encouraging, achieving 62% precision and 52% recall. These algorithms are able to learn rules similar to the ones an expert in linguistics would create, and classify candidates sorted by a confidence level.

[1] Sets of words that are cohesively grouped, but for which no deep syntactic analysis is performed.

In [15], the authors propose the use of machine learning techniques, implementing a sequential labeling algorithm based on Conditional Random Fields and a bootstrapping approach that enables the model to gradually learn linguistic particularities of the corpus. This semi supervised definition extraction tool achieved a precision of 78%, proving the advantages of using machine learning to improve definition extraction.

About evaluation, the measures usually taken are precision, recall and F1 score ([4], [6], [7], [8], [10], [11], [16], [17] amongst others). Besides this, [4] described an automatic iterative method that estimates the confidence level of a definition: the greater the number of patterns that extract a <word, definition> tuple, the higher its confidence will be, which is similar to the confidence sorting used in [11].

Some authors have tried to tackle this problem for Spanish texts as well. In [8] the authors present a system for definition extraction using pattern matching and dependency parsing that achieved a precision of 53% and a 79% recall. The authors of [10] use pattern matching for extracting definitions in a particular domain (law text) with 59% precision and 61% recall.

It is interesting to notice some difficulties reported in the papers. On one hand, [7] and [8] indicate that long definitions are harder to recognize. It could be because they span across multiple sentences, or because they often become "interrupted" by spans of text that do not belong to the definition itself. On the other hand, [8] conclude that long patterns tend to have low precision.

2.2 Crossword Generation

Crossword generation is a complex task given the wide search space it has. Building a crossword grid is a *Constraint Satisfaction Programming* task: there are layout constraints (words must begin and end in a black square or the boundary of the grid) and constraints related to the way the board has been filled up so far (previously inserted words) [12].

We analyzed different techniques that were previously used ([12], [18], [3]), all of them use the same strategy of generating the board first and then locating the words (and definitions). However, in our approach we do not start from a defined board, but we add black cells at the same time as the words, thus giving more flexibility to the process.

In [12] the system uses a priority queue that stores partial solutions sets. In each step a partial solution is dequeued and a new insertion slot is selected. The selected slot is the one that has the highest number of blank spaces, so the search space is as limited as possible. It is possible to include some randomness in order to create different results in each execution. Every time a new word is inserted, the schema is scored based on the number of completed quadrants and how likely it is to add new words in future iterations (benefit of new terms). This benefit is calculated performing a look-ahead step and counting how many definitions will be compatible in the next iteration, considering the new word and the slots it crosses.

In [18] they use Prolog predicates, exploiting the backtracking technique for word insertion. The authors do not perform the definition extraction, in this case the definitions are taken from WordNet, using relations between words to create a thematic crossword.

The authors of [3] first create the board including black squares and then they insert words using backtracking like [18], but using evolutionary algorithms for the crosswords generation. The definitions are extracted from a dictionary. Some problems with the convergence of the evolutionary algorithms implemented were reported. The authors of [19] also use an evolutionary algorithm and a wisdom of artificial crowds algorithm to generate crossword puzzles. They aim to find optimal boards that do not contain invalid words, but the use of random recombinations seems to always leave some invalid words in the crossword. In our work, on the other hand, we maintain a valid crossword puzzle at every iteration of the generation process, thus the final crossword only contains words that belong to our valid words lists.

3 System Description

Our system is divided in two modules: the first one extracts definitions from news texts, the second one creates crosswords based on the extracted definitions and a list of auxiliary words, as shown in figure 1.

The news texts are processed by the definition extraction module, which returns pairs of <definition, definendum> as output. Then, the crossword generator module uses these definitions and the ones in the external resources module to generate a crossword puzzle.

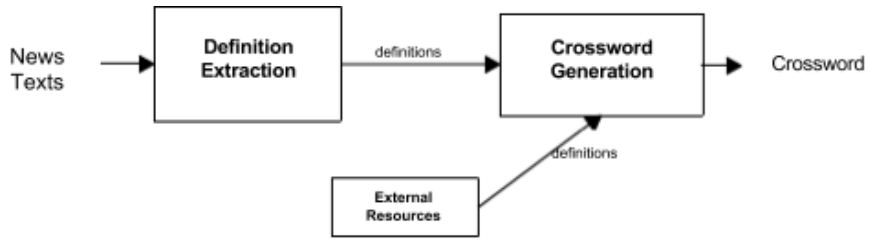

Figure 1: System design

Example

Consider the following text fragments belonging to news texts:

- *"Esta ocupación justificó la instalación de una base de la Organización del Tratado del Atlántico Norte (OTAN) en el Atlántico Sur." / "This occupation justified the installation of a base of the North Atlantic Treaty Organization (NATO) in the South Atlantic."*
- *"Las hojas son las principales consumidoras de agua dentro de la fisiología vegetal." / "Leaves are the main water consumers within plant physiology."*
- *"Por ejemplo, podemos pensar en Java como un buen lenguaje de programación porque entre otras cosas, es multiplataforma." / "For example, we can think of Java as a good programming language because, among other things, it is multi-platform."*
- *"El realizador georgiano afincado en Francia, Otar, visitó en la jornada de ayer algunos de los escenarios que, como Heleta, acogieron el rodaje de su película Euzkadi été 1982." / "The Georgian filmmaker who lives in France, Otar, visited yesterday some of the scenarios that, as Heleta, hosted the shooting of his film Euzkadi été 1982."*
- *"Según el estudio, 63 por ciento de los vendedores recibe a sus clientes con saludos poco cordiales, como «hola», «¿sí?» o «¿qué necesita?»." / "According to the study, 63 percent of sellers welcome guests with not so cordial greetings, as «hello», «yes?» or «what do you need?»"*
- *"Ana Olivera declaró «Visitante Ilustre de Montevideo» al rey de Ijero (Nigeria), Joseph Adebayo Adewole." / "Ana Olivera declared «Distinguished Visitor of Montevideo» the king of Ijero (Nigeria), Joseph Adebayo Adewole."*
- *"Un campesino contó su historia: 'yo aré durante 30 días esta tierra, la sembré, y ahora las inundaciones estropearon todo' ." / "A farmer told his story: 'I plowed for 30 days this land, planted the seed, and now floods spoiled everything' ."*
- *"Ante una infección evite compartir utensilios y tazas, lave los platos con agua caliente y jabonosa, y evite fumar en los alrededores de la vivienda." / "Dealing with an infection avoid sharing utensils and cups, wash dishes with hot, soapy water, and avoid smoking around the house."*

The definition extraction module obtains the following <definition, definendum> pairs from them:

- *<Organización del Tratado del Atlántico Norte , OTAN> / <North Atlantic Treaty Organization , NATO>*
- *<principales consumidoras de agua dentro de la fisiología vegetal , hojas> / <main water consumers within plant physiology , leaves>*
- *<buen lenguaje de programación, java> / <good programming language, java>*
- *<realizador georgiano afincado en Francia, Otar> / <Georgian filmmaker who lives in France, Otar>*
- *<saludo poco cordial, hola> / <not so cordial greeting, hello>*
- *<(_____ Olivera) declaró «Visitante Ilustre de Montevideo» al rey de Ijero (Nigeria), Ana> / <(_____ Olivera) declared «Distinguished Visitor of Montevideo» the king of Ijero (Nigeria), Ana>*

On the other hand, the external resources module offers, among others, the following definitions:

- *<voz del verbo 'arar', aré> / <form of the verb 'to plow', plowed>*
- *<voz del verbo 'lavar', lave> / <form of the verb 'to wash', wash>*

Then, using the obtained definitions, the crossword generation module builds the crossword shown in figure 2.

Figure 2: Automatically generated crossword

4 Definition Extraction

The definition extraction module receives news texts, extracts definitions from those texts, and returns them as output. The definition extraction process is shown in figure 3.

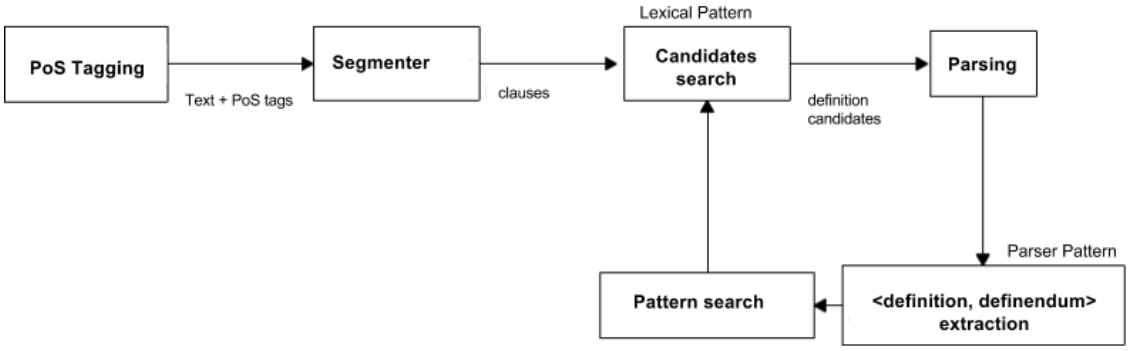

Figure 3: Extraction module design

The module contains a set of 32 manually defined patterns. We built those patterns by analyzing news texts from web sites in Spanish, and by adapting some patterns presented in [4]. The complete pattern set is shown in Appendix A.

The corpora used for pattern definitions are shown in table 1. We iteratively analysed Portal180 and MontevideoCom, used as development corpus. Initially a small set of patterns was constructed by manually inspecting both corpora, then this set was iteratively refined to recognize more definitions and improve the precision of those obtained.

After several iterations we obtained a first pattern set which was used to extract definitions from Portal180 and MontevideoCom corpora. The Portal180 corpus was manually tagged to be used as gold standard in order to evaluate the recall of the extraction process. We extracted 114 definitions from the Portal180 corpus, with a precision of 0.73 and a recall of 0.70. From the MontevideoCom corpus we extracted 474 definitions with a precision of 0.57. Because of the low precision obtained, we decided to remove and modify some low precision patterns.

Finally, in the last iteration, we extracted definitions from the held-out corpus obtaining 205 definitions with a precision of 0.69. We refined the used patterns discarding those with the lowest precision, and then we performed a new extraction evaluation. This second evaluation reached a precision of 0.75 over 172 extracted definitions.

Corpus	Size	Description
Portal180	7500	Texts extracted from the news site 180
MontevideoCom	33160	Texts extracted from the news site Montevideo Portal
Francia	21271	Texts about the same news event extracted from different news sources
Held-out	20128	Texts manually extracted from the news sites 180 and Montevideo Portal
Test	23581	Texts manually extracted El País and La República newspapers

Table 1: Corpora used for pattern definition and evaluation

The patterns consist of two parts: the *lexical pattern* and the *parser pattern*. The lexical pattern acts as a filter to obtain definition candidates from the text, while the parser pattern uses the candidate parse tree to obtain the definitions or to discard the failed candidates.

We first process the corpus with the FreeLing POS-tagger [20], which also performs a named entity recognition and classification task, and with the clause analyzer[2] ClaTex [21]. Then we search for clauses which may contain definitions by applying the lexical pattern. These clauses are processed with the FreeLing constituent parser and then <definition, definendum> pairs are extracted using the parser pattern, if possible. For each new pair a bootstrapping process is applied (as in [22]), searching for new patterns within the clauses containing both elements of the pair.

The need for these two components, the lexical pattern and the parser pattern, arises because the FreeLing parser has poor performance on long sentences. To avoid extracting wrong <definition, definendum> pairs, we limit the text to analyze to the smallest proposition containing the candidate found by the lexical pattern. It is worth noting that definitions spanning more than one sentence are beyond the scope of this work.

The last process step is the post-processing of definitions in order to give them an appropriate format as crossword clues.

4.1 Lexical Pattern

The lexical pattern is a list of tuples of the form `<word, lemma, {tag list,accepted,max}>`, where `tag list` is a POS-tag list, `accepted` states whether the words or lemmas whose POS-tags belong to the tag list must be accepted or rejected, and `max` indicates the maximum number of words to be considered. The word or the lemma could not be specified, therefore the pattern is flexible regarding the number of restrictions we can state, at the same time that the fine granularity allows us to define constraints in order to decrease the number of false positives.

For example, the fragment *"Leucemia_ Linfocítica_ Crónica (LLC)"*[3] is recognized by the lexical pattern that matches the sequence [noun, opening parentheses, noun, closing parentheses], as we can see in table 2.

Word	**Lexical Pattern** ⟨ **word**: W, **lemma**: L, {**tag list**: TL, **accepted**: A, **max**: M}⟩
Leucemia_Linfocítica_Crónica	⟨ _, _, {[Noun], TRUE, 1}⟩
(	⟨(, (, {[Parentheses], TRUE, 1}⟩
LLC	⟨_, _, {[Noun], TRUE, 1}⟩
)	⟨),), {[Parentheses], TRUE, 1}⟩

Table 2: Lexical pattern construction

[2]A clause is a linguistic unit with sentence structure, i. e., a subject-predicate structure, coordinated or subordinated to other clauses, forming a complex sentence.

[3]The POS-tagger recognizes *"Leucemia Linfocítica Crónica"* as a named entity and therefore treats it as a single word.

4.2 Parser Pattern

The parser pattern is a list of triples `<component, tag list, constituent>`, where the last two elements values depend on the first element value, as we can see in table 3.

Component	Tag list	Constituent
≺DEF≻ (definition)	tag list	constituent type
≺DFM≻ (definendum)	tag list	constituent type
≺DEFC≻ (definition complement)	-	constituent type
≺DFMC≻ (definendum complement)	-	constituent type

Table 3: Parser pattern element values

A parser pattern is a sequence of elements specifying which elements in the parse tree should be taken as definendum (≺DFM≻), definition (≺DEF≻), definendum complement (≺DFMC≻), or definition complement (≺DEFC≻).

The patterns specify the restrictions the constituents must satisfy to be identified as parts of a definition.

The definition complement is a constituent that is actually part of the phrase corresponding to the definition, but was left out by the parser. We had to treat these segments separately in order to deal with these errors in the parse tree. The definendum complement is needed for a similar reason. For instance, if we had the text *"el glaucoma es la segunda causa de ceguera en el mundo" / "glaucoma is the second cause of blindness in the world"* and these complements were not used, we would obtain the <definition, definendum> pair <*second cause of blindness*, *glaucoma*> instead of <*second cause of blindness in the world*, *glaucoma*> which is the expected definition, since it is more accurate. This situation is shown in figure 4.

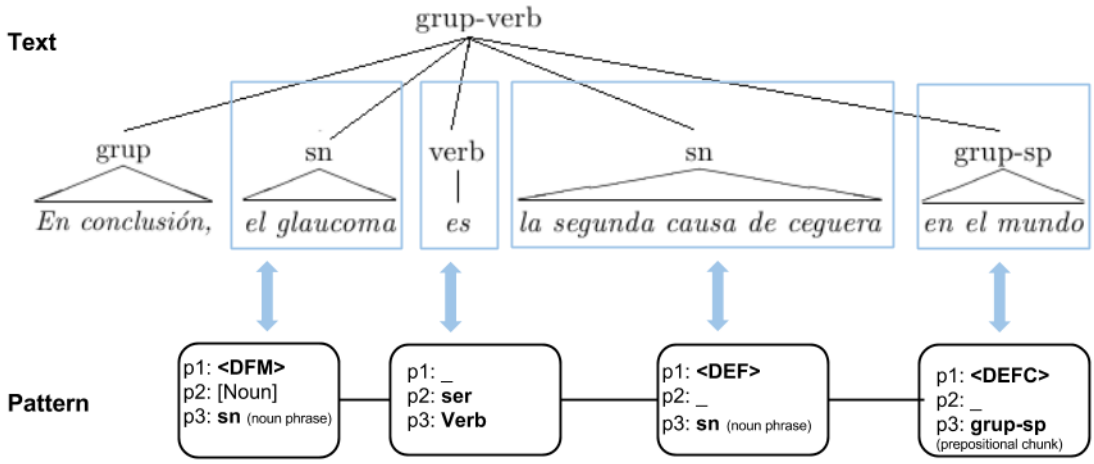

Figure 4: Correspondence between the text and the *parser pattern*.

In order to determine if the parser pattern matches the parse tree, which is represented as bracketed plain text, we use recursive regular expressions (an extension to regular expressions).

The obtained <definition, definendum> pairs are used to search for new patterns in the text. If we find a new occurrence of both elements of the pair in a sentence, we define a new pattern using the morpho-syntactic information of words between them, and the constituent types of definition and definendum.

4.3 Main Patterns

Here we present some of the defined patterns along with texts that exemplify their use. Table 4 shows two patterns using the "como" (such as) connective between definendum and definition.

Lexical Pattern	**Parser Pattern**
⟨word, lemma, {tag list, accepts, max}⟩	⟨component, tags, constituent⟩
[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨como, como, {[Conj], TRUE,1}⟩, ⟨_, _, {[Noun],FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], np⟩, ⟨como,como,Conj⟩, ⟨≺DFM≻, [Noun], coor-n|np⟩]
...especies carismáticas como los grandes felinos, los simios y los elefantes.	
(... charismatic species such as big cats, apes and elephants).	
[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[Punto], FALSE,5}⟩, ⟨_, _, {[Comma], TRUE,1}⟩, ⟨como, como, {[Conj], TRUE,1}⟩, ⟨_, _, {[Noun],FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], np⟩, ⟨_,_,Comma⟩, ⟨como,como,Conj⟩, ⟨≺DFM≻, [Noun], np⟩]
...otros cultivos, como el frijol de primera cosecha...	
(...other crops such as first crop beans...)	

Table 4: Patterns connected with "como"

Table 5 shows two patterns based on verbs.

Lexical Pattern	**Parser Pattern**
⟨word, lemma, {tag list, accepts, max}⟩	⟨component, tags, constituent⟩
[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[Period], FALSE,5}⟩, ⟨_, ser, {[Verb], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻,[Proper Noun], np⟩, ⟨≺DFMC≻,_, grup-sp⟩, ⟨_, ser, Verbo⟩, ⟨≺DEF≻, [Noun], np⟩, ⟨≺DEFC≻,_, grup-sp⟩]
El glaucoma es la segunda causa de ceguera en el mundo.	
(Glaucoma is the second cause of blindness in the world.)	
[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verbo], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	⟨≺DFM≻,[Proper Noun], np⟩, ⟨≺DFMC≻,_, grup⟩, ⟨_,_,Verbo⟩, ⟨≺DEF≻, _, coor-n|np⟩, ⟨≺DEFC≻,_, grup|np⟩]
Mujica respaldó importante inversión minera.	
(Mujica supported important mining investment.)	

Table 5: Verb based patterns

Lexical Pattern	**Parser Pattern**
⟨word, lemma, {tag list, accepts, max}⟩	⟨component, tags, constituent⟩
[⟨_, _, {[Noun], TRUE,1}⟩,	
⟨_,_, {[Noun], FALSE,5}⟩,	[⟨≺DEF≻, _, np⟩,
⟨(, (, {[Parenthesis], TRUE,1}⟩,	⟨(, (, Parenthesis⟩,
⟨_, _, {[Noun], TRUE,1}⟩,	⟨≺DFM≻,[Named entity],noun-phrase⟩,
⟨),), {[Parenthesis], TRUE,1}⟩]	⟨),), Parenthesis⟩]

Table 6: Pattern example

4.4 Complete Extraction Example

Consider the pattern shown in table 6 and the following text fragment from our corpus.

"En conclusión, la Administración de Servicios de Salud del Estado (ASSE) es el prestador estatal de salud pública en Uruguay." / *"In conclusion, the State Health Services Administration (SHSA) is the public health state provider in Uruguay."*

First, the system detects a matching between the text, previously tagged by FreeLing and segmented in clauses by Clatex, and the lexical pattern, obtaining a clause which may contain a definition, as figure 5 shows.

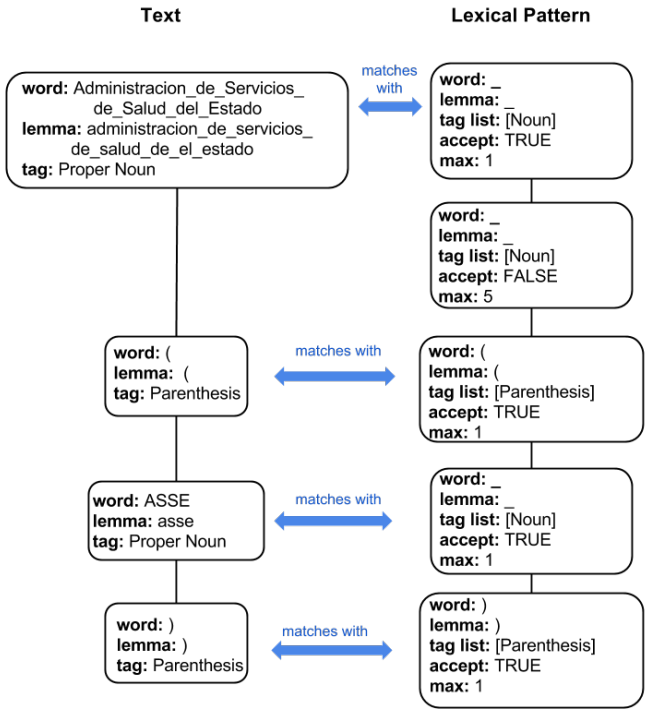

Figure 5: Correspondence between the text and the *lexical pattern*

In a second step, the candidate clause is parsed in order to find coincidences between its parsing tree and the corresponding parser pattern. As we can see in figure 6, the system generates the pair <*administración de servicios de salud del estado, ASSE*>.

Finally, the system tries to discover new patterns from the found <definition, definendum> pair, searching for clauses containing both elements of the pair. For example, from the following text fragment:

"The State Health Services Administration, known by its acronym SHSA, has a service network all over the country."

We should find the new pattern shown in the table 7.

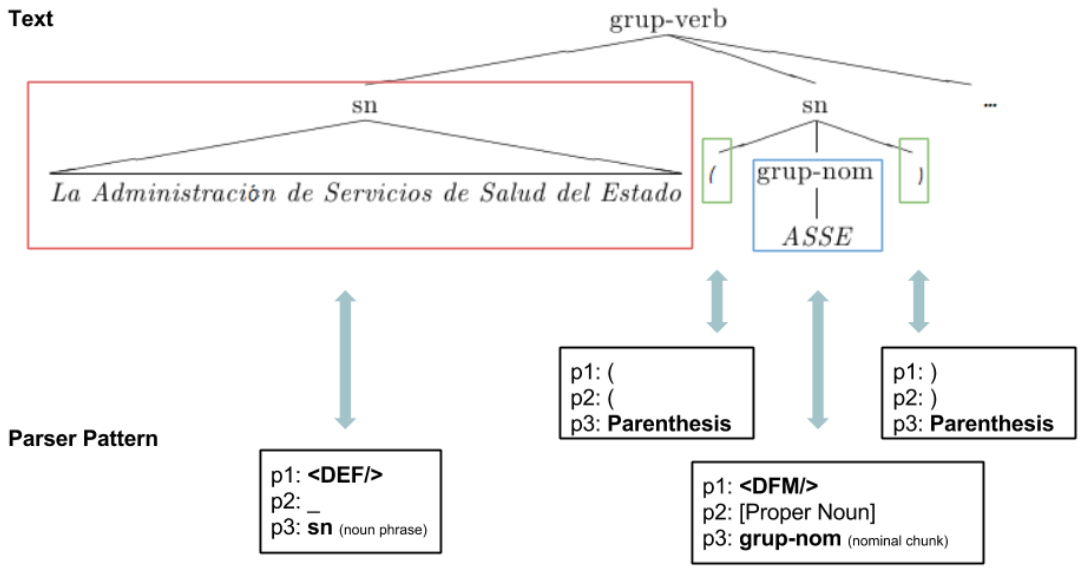

Figure 6: Correspondence between the text and the *parser pattern*

Lexical Pattern	**Parser Pattern**
⟨word, lemma, {tag list, accepts, max}⟩	⟨component, tags, constituent⟩
	[⟨≺DEF≻, _, np⟩,
[⟨_, _, {[Comma], TRUE,1}⟩,	⟨_, _,Comma⟩,
⟨_,know, {[Verb], TRUE,1}⟩,	⟨_,know,Verb⟩,
⟨_, by, {[Preposition], TRUE,1}⟩,	⟨_, by,Preposition⟩,
⟨_, its, {[Determiner], TRUE,1}⟩,	⟨_, its,Determiner⟩,
⟨_, acronym, {[Noun], TRUE,1}⟩]	⟨_, acronym,Noun⟩,
	⟨≺DFM≻,[Named entity],noun-phrase⟩]

Table 7: New extracted pattern

4.5 Definition Post-Processing

Once the definition extraction process is complete, the system post-processes the extracted pairs discarding those not useful and formatting the remaining to give them a crossword definition appearance.

At this stage, several post-processing rules are applied: the pairs which are context dependent are discarded, definendums containing too frequent words are also discarded, some templates for definition formatting are applied, and, finally, <clue, word> pairs are generated. Note that each <definition, definendum> pair can lead to more than one <clue, word> pair. This happens when the definendum contains more than one word and all of them are appropriate to be used in the crossword.

For instance, if we extracted the pair <*Ricardo Erlich, Montevideo*>, we use the named entity classification module included in the FreeLing POS-tagger to obtain the class of both named entities, which are Person and Location, respectively. For this class combination we apply a template generating the following <clue, word> pair: <*Lugar al que pertenece Ricardo Erlich, Montevideo*> / <*Place where Ricardo Erlich is from, Montevideo*>.

4.6 Extraction Evaluation

Using the final pattern set we applied the definition extraction process on the test corpus, containing texts from El País and La República newspapers (23581 words). This evaluation reached a precision of 0.73 (159 true positives and 59 false positives). From these 218 extracted <definendum, definition> pairs, 410 <clue, word> pairs were generated.

Figure 7 shows the precision reached by each defined pattern which extracted definitions from the test corpus. Figure 8 shows the number of correct and wrong definitions extracted by each pattern.

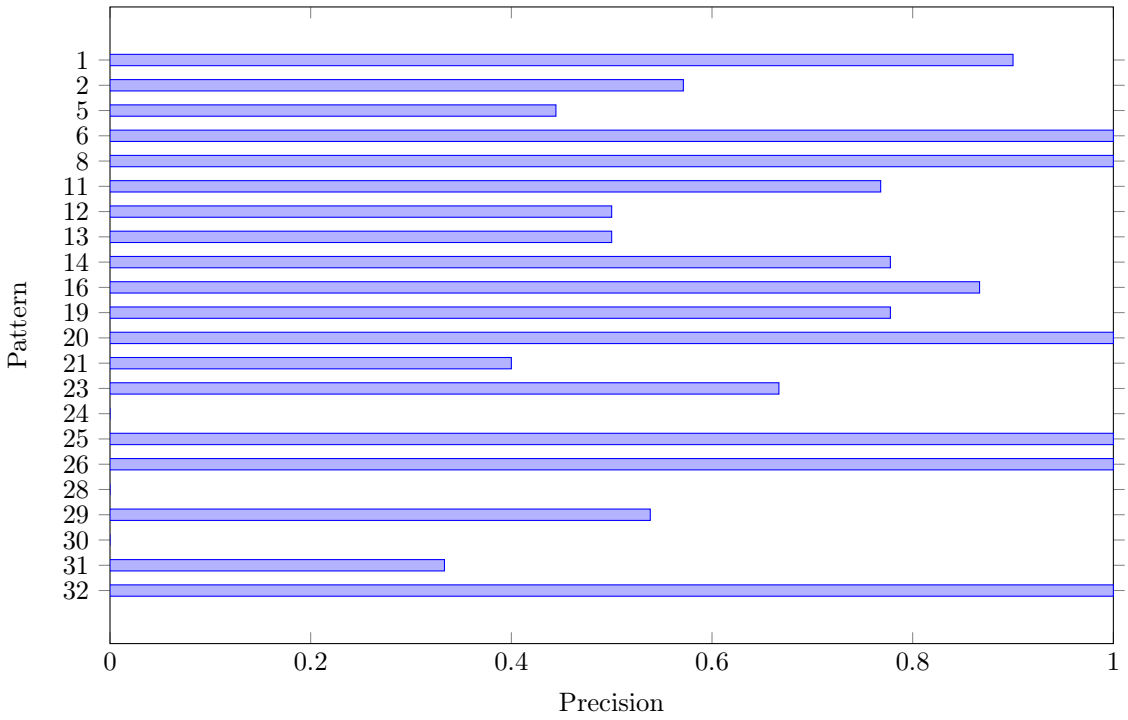

Figure 7: Precision of patterns which extracted definitions

It can be observed that high precision patterns extract most of the definitions, while low precision patterns only retrieved a few, causing little impact on the system's overall precision.

Some correct extracted definitions are shown in table 8.

Word	**Clue**
IBGE	*Instituto Brasileño de Geografía y Estadística / Brazilian Institute of Geography and Statistics*
Judith	*(_____ Uturbey). La subdirectora del Hospital de Ojos / Assistant Director of the Eye Hospital*
Puntigliano	*Ex presidente de la Administración Nacional de Puertos / Former President of the National Port Administration*
Vaticano	*(El _____). Organización que defiende el celibato pese a críticas por casos de pedofilia / Organization defending celibacy despite criticism arising from pedophile cases*
Uruguay	*Lugar que obtendrá 80 millones del Focem a través de un préstamo / Place that will get 80 millions from Focem through a loan*
Glaucoma	*Segunda causa de ceguera en el mundo / Second cause of blindness in the world*
Auto	*Un carro / A car*
Tokio	*Segunda plaza financiera del planeta / Second financial center of the planet*

Table 8: Correct extracted definitions

Some of the wrong extracted definitions can be seen in table 9.

Word	**Clue**
Canelones	*Organización a la que pertenece Marcos Carámbula / Organisation to which Marcos Carámbula belongs*
CNT	*Organización relacionada con los dirigentes de la Convención Nacional de Trabajadores / Organisation related to the leaders of the National Workers Convention*
Libia	*Los países vecinos / The neighbouring countries*
Honduras	*Laos y Camboya*

Table 9: Wrong extracted definitions

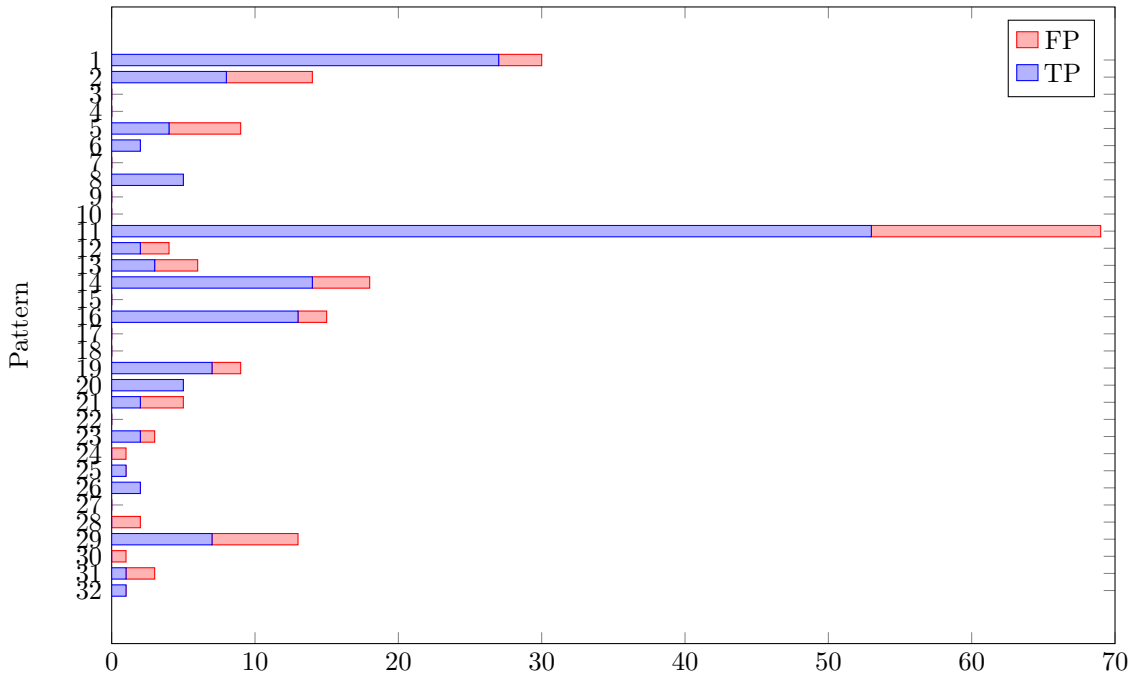

Figure 8: Number of correct and wrong definitions extracted by each pattern

Number	Lexical Pattern	Parser Pattern
1	[⟨_, _, {[Noun], TRUE,1}⟩,	
	⟨_,_, {[Noun], FALSE,5}⟩,	[⟨≺DEF≻, _, np⟩,
	⟨(, (, {[Parenthesis], TRUE,1}⟩,	⟨(, (, Parenthesis⟩,
	⟨_, _, {[Noun], TRUE,1}⟩,	⟨≺DFM≻,[Named entity],noun-phrase⟩,
	⟨),), {[Parenthesis], TRUE,1}⟩]	⟨),), Parenthesis⟩]
11	[⟨_, _, {[Named entity], TRUE,1}⟩,	[⟨≺DFM≻,[Named entity], np⟩,
	⟨_, _, {[Noun], FALSE,5}⟩,	⟨≺DFMC≻,_, phrase⟩,
	⟨_, _, {[Verb], TRUE,1}⟩,	⟨_,_,Verb⟩,
	⟨_, _, {[Noun], FALSE,5}⟩,	⟨≺DEF≻, _, np⟩,
	⟨_, _, {[Noun], TRUE,1}⟩]	⟨≺DEFC≻,_, phrase⟩]

Table 10: Patterns extracting most definitions

In the first example, the definition for the word *Canelones* is wrong because the word refers to a place and not to an organisation. The problem was caused by the named entity classification module included in the FreeLing POS-tagger. In the second example, the definition should be just *"Convención Nacional de Trabajadores"*, corresponding to the acronym *"CNT"*. In the third example, the definition lacks context to be interpreted. In the last case, the used constituent parser (FreeLing) failed in the coordination analysis. We note that the last three errors were caused by parser failures.

As figure 8 shows, patterns 1 and 11 extracted more definitions than the others, reaching precision values of 0.9 and 0.7 respectively. Table 10 shows these two patterns.

4.7 Pattern Bootstrapping

As mentioned above, after extracting <definition, definendum> pairs from the corpus by applying a pattern set, we try to automatically find new patterns matching the found pairs. This bootstrapping stage did not produce good results, since no new relevant patterns were detected.

We analysed different factors which could explain these negative results.

A first factor that could explain the bootstrapping failure was the presence of implementation bugs. We applied the procedure on a manually created example set, specifically designed to test the implementation. For these artificial examples the pattern bootstrapping method was able to identify correct new patterns. For instance, for the text fragments *"El hombre es un animal racional." / "The man is a rational being."* and *"El hombre es conocido como un animal racional." / "The man is known as a rational being."*, the

system could extract the pattern *"es conocido como" / "is known as"* from the <definition, definendum> pair extracted by the pattern *"es" / "is"*.

Once implementation issues were ruled out, another possible explanation for the negative bootstrapping results is that the nature of the used texts is not adequate for this task since, once a term is defined in the corpus, it is not defined again in other documents from the corpus. To prove this hypothesis we constructed a new corpus containing texts from different sources, all of them about the same topic. This new corpus (Francia corpus, with 21271 words) includes news about the Charlie Hebdo attack on January 2015. Since all the texts talk about the same news event, we assumed there would be several occurrences of each term, increasing the probability of finding new patterns for the <definition, definendum> pairs extracted. However, this did not happen, and it was not possible to find new patterns in the bootstrapping stage.

We believe that new pattern identification from previous <definition, definendum> pairs may be possible if we work with a significantly larger amount of text for increasing the likelihood of finding repeated term definitions. It is also possible to increase the abstraction level of definitions before looking for new patterns.

5 Crossword Generation

This section describes the implemented process for building a crossboard grid using the extracted definitions. We also show a complete execution example.

5.1 Generation Process

The crossword generation module is based on a greedy algorithm written in Prolog. Due to the nature of the greedy algorithm, we make sure that at the end of each step of the process the crossword that is being built is consistent. This means that if we stopped the process after any step and replaced all blank spaces in the grid with black squares, the result would be a valid crossword. The use of Prolog, with its built-in unification mechanism, made the code simpler and the constraint checking more efficient.

The generation module receives as input the words extracted as definitions by the extraction module as well as other words retrieved from external resources (see section 5.2). All these words are divided in three sets:

1. *uniword*: words extracted from a definendum that contains only one word. For example: *<Instituto Interamericano de Derechos Humanos, IIDH> / <Interamerican Institute for Human Rights, IIHR>*
2. *multiword*: words extracted from a definendum that contains more than one word. For example: *<(Ana _____) Intendente de Montevideo, Olivera> / <(Ana _____) Intendant of Montevideo, Olivera>*
3. *external*: words obtained from an external resource. For example: *<Apócope de Nueva York, NY> / <Short for New York, NY>*

The division between uniwords and multiwords allows us to control the quality of the clues we are including in the final crossword. Multiword definitions imply showing a blank space in the clue (such as *"Ana _____"* in the above example) that has to be filled by the user. Having too many of this type of clues in a crossword makes it less elegant and less attractive for the user.

The algorithm calculates a score for each candidate word, i.e. a word that might be located in an empty slot inside the crossword grid. This score takes into account:

- How many words already on the grid are intersected by this new word.
- How many new words are created on the grid due to intersections. All the new words must be valid, which means they must belong to one of the sets of words.

These are the steps of the algorithm:

Step 4 implies selecting between one of the three sets of words. If it is possible to use the *uniword* or *multiword* sets, one of them is randomly chosen with certain probability, being the likelihood of the *uniword* set higher. If it is no longer possible to add more words from any of these sets, the *external* set is used.

5.2 External Resources Module

The external resources module is a <clue, word> pair list. The goal of this module is to contribute with more words to generate valid crosswords, with as few black squares as possible, once the automatically extracted word set is exhausted. The list was manually built using definitions from the Spanish WordNet [23], a list of Uruguayan terms[4] and a list of two and three letter words[5].

[4] http://www.mec.gub.uy/academiadeletras/Bpalabras/Pp_Palabras.htm
[5] http://www.scrabbel.org.uy/ayuda/dos_y_tres_letras.htm

Algorithm 1 Greedy crossword generation algorithm

```
Pick an initial word
while it is possible to add new words do
    Switch direction (horizontal or vertical)
    Choose a set of candidate words
    for each word in the set do
        for each available position do
            Calculate score for this word and position
    Locate the word with highest score in the grid
Set all remaining empty cells in the grid as black squares
```

5.3 Algorithm Execution Example

Consider the following sets of words:

1. *uniword*: {ZINC, IIDH, INIA, ARCO}
2. *multiword*: {CO, INTERNET, ANA, OLIVERA}
3. *external*: {AYO, NY, . . . }

The algorithm proceeds as shown in figure 9. First it randomly chooses the word *"zinc"* and locates it in the grid. During the first iteration, the process will choose the word from the *uniword* set that has the highest score. Having a high score means it will have to intersect the current words in the grid. At this point there is more than one possible word with the highest score: both *"INIA"* and *"arco"* have the same score. In this case *"INIA"* is selected. The next iteration will try to locate a horizontal word from the *uniword* or *multiword* sets, so the only option that does not introduce invalid words is *"IIDH"*. After this step, there are no more words from either the *uniword* or the *multiword* sets that are consistent with the crossword, so from now on the *external* set is used. Once now more words can be added without making the crossword inconsistent, the remaining blank cells are replaced with black squares to complete the crossword.

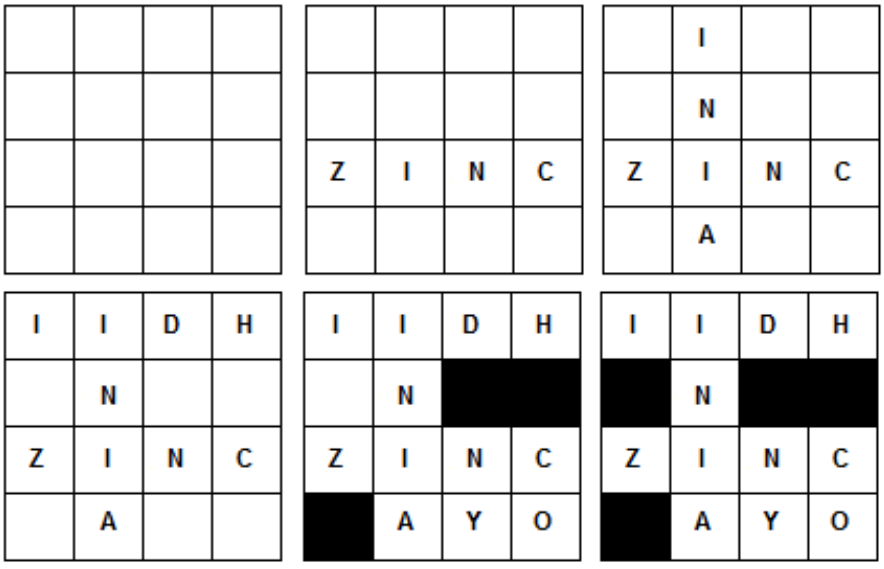

Figure 9: Crossword generation algorithm execution example

We empirically observed that after this process around 50% of the selected words that end up composing the crossword, on average, correspond to words extracted by the extraction module, while the other 50% are words obtained from the external resources. The external resources words are in general short words used to fill small gaps that would be very difficult to fill with extracted words.

6 Conclusions and future work

We built a system that addresses the complete process of crossword generation, taking natural language texts in Spanish and generating crosswords in a fully automatic way. The system starts from a collection of texts and performs a definition extraction process, followed by the construction of a crossword from these definitions. This could be applied to any collection of texts to build themed crosswords based on them.

The definition extraction system reaches 73% precision in the testing corpus. Comparing the results obtained with the current literature, this result seems very promising. Similar works for Spanish texts, like [8] and [10], where the authors also use pattern matching, achieved precisions of 53% and 59% respectively.

On the other hand, researchers that applied Machine Learning techniques to tackle the same problem ([15]), achieved a precision of 78%, so it seems that we are headed in the right direction, but there is still room for improvement.

The crossword generation system builds crosswords similar to those built by humans, in that the occurrence of longer words and greater number of intersections are prioritized. About half of the words used come from the external resources module (short words in general) and half from the extraction module.

Among the difficulties we found, the ones that stand out are that the pattern *bootstrapping* did not work, and that the lack of quality tools for Spanish forced us to make adjustments in the algorithms to try to reduce the error rate, but this still has a negative impact on the system.

As future work, it would be desirable to improve the base patterns, increasing the *recall* and *precision*. The definitions could be categorized to enhance their recognition, and we could exploit the analysis of semantic relations and syntactic functions to find more definitions. An interesting point to work is to evaluate the use of *machine learning* techniques rather than a rule-based approach, or a combination of both. About the generation process, although the greedy algorithm produces satisfactory results, it would be interesting to try out some other strategies that might yield better crosswords. For example we could try a beam search approach that explores a set of promising partial crossword grids and returns the one that uses the fewest number of black squares, thus maximizing the space used by words in the grid. It would also be desirable to explore the possibility of generating thematic crosswords for educational purposes, possibly working on other types of corpus.

References

[1] S. Walter and M. Pinkal, "Automatic extraction of definitions from German court decisions," in *Proceedings of the workshop on information extraction beyond the document*. Association for Computational Linguistics, 2006, pp. 20–28.

[2] L. Rigutini, M. Diligenti, M. Maggini, and M. Gori, "A fully automatic crossword generator," in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 2008, pp. 362–367. [Online]. Available: https://doi.org/10.1109/icmla.2008.104

[3] J. Engel, M. Holzer, O. Ruepp, and F. Sehnke, "On computer integrated rationalized crossword puzzle manufacturing," in *Fun with Algorithms*. Springer, 2012, pp. 131–141. [Online]. Available: https://doi.org/10.1007/978-3-642-30347-0_15

[4] R. M. Ortega Mendoza, "Descubrimiento automático de hipónimos a partir de texto no estructurado," Master's thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, México, 2007.

[5] A. Przepiórkowski, Ł. Degórski, B. Wójtowicz, M. Spousta, V. Kuboň, K. Simov, P. Osenova, and L. Lemnitzer, "Towards the automatic extraction of definitions in Slavic," in *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*. Association for Computational Linguistics, 2007, pp. 43–50.

[6] S. Muresan and J. Klavans, "A method for automatically building and evaluating dictionary resources," in *Proceedings of the Language Resources and Evaluation Conference*, 2002.

[7] R. Del Gaudio and A. Branco, "Automatic extraction of definitions in Portuguese: A rule-based approach," in *Progress in artificial intelligence*. Springer, 2007, pp. 659–670. [Online]. Available: https://doi.org/10.1007/978-3-540-77002-2_55

[8] G. Sierra, "Extracción de contextos definitorios en textos de especialidad a partir del reconocimiento de patrones lingüísticos," *Linguamática*, vol. 1, no. 2, pp. 13–37, 2009.

[9] L. Rigutini, M. Diligenti, M. Maggini, and M. Gori, "Automatic generation of crossword puzzles," *International Journal on Artificial Intelligence Tools*, vol. 21, no. 03, 2012. [Online]. Available: https://doi.org/10.1142/S0218213012500145

[10] A. Sánchez and M. Márquez, "Hacia un sistema de extracción de definiciones en textos jurídicos," *Actas de la 1er Jornada Venezolana de Investigación en Lingüística e Informática*, pp. 1–10, 2005.

[11] C. Borg, M. Rosner, and G. Pace, "Evolutionary algorithms for definition extraction," in *Proceedings of the 1st Workshop on Definition Extraction*. Association for Computational Linguistics, 2009, pp. 26–32.

[12] B. Ranaivo-Malançon, T. Lim, J.-L. Minoi, and A. J. R. Jupit, "Automatic generation of fill-in clues and answers from raw texts for crosswords," in *Information Technology in Asia (CITA), 2013 8th International Conference on*. IEEE, 2013, pp. 1–5. [Online]. Available: https://doi.org/10.1109/CITA.2013.6637572

[13] L. Degórski, M. Marcińczuk, and A. Przepiórkowski, "Definition extraction using a sequential combination of baseline grammars and machine learning classifiers," in *LREC*. Citeseer, 2008.

[14] G. Sierra, "Extracción de relaciones léxicas para dominios restringidos a partir de contextos definitorios en español," UNAM, IPN, INAOE, COLMEX, UAM, México, Tech. Rep., 2011.

[15] L. E. Anke, R. Carlini, H. Saggion, and F. Ronzano, "Defext: A semi supervised definition extraction tool," *CoRR*, 2016.

[16] S. Harabagiu, C. A. Bejan, and P. Morărescu, "Shallow semantics for relation extraction," in *Proceedings of the 19th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., 2005, pp. 1061–1066.

[17] N. Bach and S. Badaskar, "A review of relation extraction," *Literature review for Language and Statistics II*, 2007.

[18] A. Aherne and C. Vogel, "Crossing wordnet with crosswords, netting enhanced automatic crossword generation," TCD-CS-2005-52, Tech. Rep., 2005.

[19] D. Bonomo, A. P. Lauf, and R. Yampolskiy, "A crossword puzzle generator using genetic algorithms with wisdom of artificial crowds," in *Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), 2015*. IEEE, 2015, pp. 44–49. [Online]. Available: https://doi.org/10.1109/CGames.2015.7272960

[20] L. Padró, M. Collado, S. Reese, M. Lloberes, and I. Castellón, "Freeling 2.1: Five years of open-source language processing tools," in *7th International Conference on Language Resources and Evaluation*, 2010. [Online]. Available: https://doi.org/10.1145/2738046

[21] D. Wonsever, S. Caviglia, J. Couto, and A. Rosá, "Un sistema para la segmentación en proposiciones de textos en Español," *Letras de hoje*, vol. 144, p. 41, 2006.

[22] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, pp. 539–545. [Online]. Available: https://doi.org/10.3115/992133.992154

[23] A. G. Agirre, E. Laparra, G. Rigau, and B. C. Donostia, "Multilingual central repository version 3.0: upgrading a very large lexical knowledge base," in *GWC 2012 6th International Global Wordnet Conference*, 2012, p. 118.

Appendix A: Patterns

The following are the 32 base patterns used by the definition extraction module.

Number	Lexical Pattern	Parser Pattern
1	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨(, (, {[Parenthesis], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨),), {[Parenthesis], TRUE,1}⟩]	[⟨≺DEF≻, _, sn⟩, ⟨(, (, Parenthesis⟩, ⟨≺DFM≻,[Proper Noun], nominal chunk⟩, ⟨),), Parenthesis⟩]
2	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨_, ,, {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[Comma], TRUE,1}⟩]	[⟨≺DEF≻, _, sn⟩, ⟨≺DFM≻,[Proper Noun], nominal chunk⟩, ⟨ , , Comma⟩]
3	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨(, (, {[Parenthesis], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨),), {[Parenthesis], TRUE,1}⟩]	[⟨≺DFM≻, [Noun], nominal chunk⟩, ⟨(, (, Parenthesis⟩, ⟨≺DEF≻, _, nominal chunk⟩, ⟨),), Parenthesis⟩]
4	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨(, (, {[Parenthesis], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨),), {[Parenthesis], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨(, (, Parenthesis⟩, ⟨≺DFM≻, [Noun], nominal chunk⟩, ⟨),), Parenthesis⟩]
5	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨ 'es', 'ser', {[Verb], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻,[Noun], noun phrase⟩, ⟨≺DEFC≻,_, prepositional chunk⟩, ⟨_, 'ser',Verb⟩, ⟨≺DFM≻, [Noun], noun phrase | coordinating conjunction⟩]
6	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨, , {[Comma], TRUE,1}⟩]	[⟨≺DEF≻, _, noun phrase | coordinating conjunction⟩, ⟨≺DEFC≻, _,prepositional chunk⟩, ⟨≺DFM≻,[Proper Noun],nominal chunk⟩, ⟨ , , Comma⟩]
7	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨, , {[Comma], TRUE,1}⟩]	[⟨≺DEF≻, _, noun phrase⟩, ⟨≺DFM≻,[Proper Noun],nominal chunk⟩, ⟨ , , Comma⟩]
8	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨_, 'ser', {[Verb], TRUE,1}⟩, ⟨_, _, {[All], FALSE,10}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻,[Proper Noun], noun phrase ⟩, ⟨≺DFMC≻,_, prepositional chunk⟩, ⟨_, 'ser',Verb⟩, ⟨≺DEF≻, [Noun], noun phrase⟩,
9	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨, , {[Comma], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨ , , Comma⟩, ⟨≺DEF≻, _,nominal chunk⟩, ⟨ , , Comma⟩]

Number	Lexical Pattern	Parser Pattern
10	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[All], TRUE,10}⟩, ⟨, , {[Hyphen], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[All], TRUE,10}⟩, ⟨, , {[Hyphen], TRUE,1}⟩]	[⟨≺DFM≻, [Noun], nominal chunk⟩, ⟨, , Hyphen⟩] ⟨≺DEF≻,[Noun],noun phrase⟩, ⟨, , Hyphen⟩]
11	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻,[Proper Noun], noun phrase⟩, ⟨≺DFMC≻,_,chunk⟩, ⟨_,_,Verb⟩, ⟨≺DEF≻, _, noun phrase⟩, ⟨≺DEFC≻,_, chunk⟩]
12	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨_, _, {[Verb], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase | nominal chunk⟩, ⟨, , Comma⟩, ⟨≺DEF≻,[Noun],relative subordinate⟩]
13	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨., ., {[Period], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨≺DFM≻,[Proper Noun],noun phrase⟩, ⟨, , Comma⟩]
14	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨'como', 'como', {[CS], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨'como', 'como', subordinating conjunction⟩, ⟨≺DFM≻,[Noun],noun phrase⟩]
15	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[All], TRUE,10}⟩, ⟨–, –, {[Dash], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[All], TRUE,10}⟩, ⟨–, –, {[Dash], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨–, –, Dash⟩] ⟨≺DFM≻,[Noun],noun phrase⟩]
16	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻,[Noun], noun phrase⟩, ⟨≺DFMC≻,_, coordinated nominal chunk⟩, ⟨_,_,Verb | adverb⟩, ⟨_,_,Verb⟩, ⟨≺DEF≻, [Noun], noun phrase | prepositional chunk⟩, ⟨≺DEFC≻,_, prepositional phrase⟩]
17	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨., ., {[Period], TRUE,1}⟩]	[⟨≺DFM≻, [Noun], coordinated nominal chunk⟩, ⟨, , Comma⟩, ⟨≺DEF≻, _, adjective phrase⟩, ⟨≺DEFC≻,_, prepositional chunk⟩]
18	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨., ., {[Period], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨, , Comma⟩, ⟨≺DEF≻, [Noun], adjective phrase⟩, ⟨≺DEFC≻,_, prepositional chunk | prepositional phrase⟩]
19	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Noun], noun phrase⟩, ⟨≺DFMC≻,_, noun phrase⟩, ⟨_, _, Verb⟩, ⟨≺DEF≻, [Determiner], noun phrase⟩, ⟨≺DEFC≻,_, chunk⟩]
20	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨., ., {[Period], TRUE,1}⟩]	[⟨≺DEF≻, [det], noun phrase⟩, ⟨≺DEFC≻,_, prepositional chunk⟩, ⟨, , Comma⟩, ⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨≺DFMC≻,_, prepositional chunk⟩, ⟨., ., Period⟩]
21	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨_, _, {[Comma], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨≺DFMC≻,_, noun phrase⟩, ⟨, , Comma⟩, ⟨≺DEF≻, [Noun], noun phrase⟩]
22	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨:, :, {[Colon], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨≺DEFC≻,_, prepositional chunk⟩, ⟨:, :, Colon⟩, ⟨≺DFM≻, [Noun], coordinated nominal chunk⟩]

Number	Lexical Pattern	Parser Pattern
23	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[All], TRUE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨'como', 'como', {[CS], TRUE,1}⟩, ⟨_, _, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, [Noun], noun phrase⟩, ⟨, , Comma⟩, ⟨'como', 'como', subordinating conjunction⟩, ⟨≺DFM≻,[Noun],noun phrase⟩]
24	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Noun], noun phrase⟩, ⟨_, _, adverb⟩, ⟨≺DEF≻, [Determiner], subordinating part⟩]
25	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨'se', 'se', {[Pronoun], TRUE,1}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨'se', 'se', Pronoun⟩, ⟨≺DEF≻, [Noun], grup-verb⟩, ⟨≺DEFC≻, _, prepositional phrase⟩]
26	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨≺DFMC≻, _, 'de' chunk⟩, ⟨_, _, Verb⟩, ⟨≺DEF≻, _, prepositional phrase⟩, ⟨≺DEFC≻, _, prepositional chunk⟩]
27	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨'conocido', 'conocer', {[Verb], TRUE,10}⟩, ⟨'como', 'como', {[CS], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DEF≻, _, noun phrase⟩, ⟨'conocido', 'conocer', Verb⟩, ⟨'como', 'como', subordinating conjunction⟩, ⟨≺DFM≻,_,noun phrase⟩]
28	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Noun, Verb], noun phrase⟩, ⟨_, _, adverb⟩, ⟨≺DEF≻, [Noun], chunk⟩]
29	[⟨_, _, {[Determiner], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_, _, {[All], TRUE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨, , {[Comma], TRUE,1}⟩]	[⟨≺DEF≻, _, noun phrase⟩, ⟨, , Comma⟩] ⟨≺DFM≻,[Proper Noun],nominal chunk⟩, ⟨, , Comma⟩]
30	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨., ., {[Period], TRUE,1}⟩]	[⟨≺DEF≻, _, noun phrase⟩, ⟨≺DEFC≻,_, prepositional chunk⟩, ⟨, , Comma⟩, ⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨≺DFMC≻, _, prepositional chunk⟩]
31	[⟨_, _, {[Proper Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨, , {[Comma], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Period], FALSE,10}⟩, ⟨_, _, {[Comma], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨≺DEF≻, [Comma], coordinated nominal chunk⟩, ⟨≺DEFC≻, _, noun phrase⟩}]
32	[⟨_, _, {[Noun], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Verb], TRUE,1}⟩, ⟨_,_, {[Noun], FALSE,5}⟩, ⟨_, _, {[Noun], TRUE,1}⟩]	[⟨≺DFM≻, [Proper Noun], noun phrase⟩, ⟨_, _, Verb⟩, ⟨≺DEF≻, [Noun], infinitive whose head is prepositional⟩]