Construcción de Recursos Lingüísticos para una Gramática HPSG para el Español

Luis Chiruzzo
luischir@fing.edu.uy

Diciembre de 2015

Supervisora: Dina Wonsever
wonsever@fing.edu.uy
Instituto de Computación, Facultad de Ingeniería
Universidad de la República

Tribunal: Laura Alonso Alemany, Aiala Rosá, Javier Baliosian

Tesis de Maestría en Computación

PEDECIBA Informática
Instituto de Computación - Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Resumen

En este trabajo se presenta la construcción de recursos lingüísticos para trabajar con una gramática HPSG para el español. HPSG es un formalismo gramatical rico debido a que el resultado del análisis sintáctico con este formalismo es una representación de la oración que incluye información tanto sintáctica como semántica. Para el idioma inglés existen *parsers* estadísticos HPSG de alta performance y cobertura del idioma, pero para el español las herramientas existentes aún no llegan al mismo nivel.

Se describe una gramática HPSG para el español, indicando sus estructuras de rasgos principales y sus reglas de combinación de expresiones. Se construyó un corpus de árboles HPSG para el español utilizando la gramática definida. Para esto, se partió del corpus AnCora y se transformaron las oraciones mediante un proceso automático, obteniendo como resultado un nuevo corpus etiquetado según el formalismo HPSG. Las heurísticas de transformación tienen un 95,3% de precisión en detección de núcleos y un 92,5% de precisión en clasificación de argumentos.

A partir del corpus se definieron las entradas léxicas y se agruparon las entradas de las categorías léxicas de mayor complejidad combinatoria (verbos, nombres y adjetivos) según su comportamiento sintáctico-semántico. Estas agrupaciones de entradas léxicas se denominan *frames léxicos*. A partir de esto se construyó un *supertagger* para identificar los *frames léxicos* más probables dadas las palabras de una oración. El supertagger tiene un *accuracy* de 83,58% para verbos, 85,78% para nombres y 81,40% para adjetivos (considerando las tres etiquetas más probables).

Palabras clave: HPSG, español, corpus, parsing, supertagging.

Contenido

1 Introducción **5**
- 1.1 Motivación . . . 5
- 1.2 Etapas para la construcción de un parser . . . 6
- 1.3 Objetivos . . . 7
- 1.4 Estructura del documento . . . 7

2 Antecedentes **9**
- 2.1 Formalismos gramaticales . . . 9
- 2.2 HPSG . . . 11
  - 2.2.1 DELPH-IN . . . 12
  - 2.2.2 Spanish Resource Grammar . . . 12
  - 2.2.3 Enju . . . 18
- 2.3 AnCora . . . 22

3 Gramática propuesta **25**
- 3.1 Fundamentos . . . 25
- 3.2 Estructuras de rasgos . . . 28
  - 3.2.1 Palabra . . . 28
  - 3.2.2 Frase . . . 31
- 3.3 Reglas de la gramática . . . 32
  - 3.3.1 Reglas de especificador . . . 32
  - 3.3.2 Reglas de complemento . . . 32
  - 3.3.3 Reglas de modificador . . . 37
  - 3.3.4 Coordinaciones . . . 38
  - 3.3.5 Reglas simplificadas . . . 40

4 Transformación del corpus **45**
- 4.1 Proceso de transformación . . . 45
- 4.2 Estructura de AnCora . . . 45
- 4.3 Proceso top-down . . . 47
  - 4.3.1 Marcado de clíticos . . . 47
  - 4.3.2 Extracción de bloques . . . 48
  - 4.3.3 Extracción de puntuación . . . 49

4.3.4 Sintagmas nominales que actúan como expresiones temporales . . . 50
4.3.5 Incisos . . . 50
4.3.6 Subordinadas y conjunciones al inicio . . . 52
4.3.7 Relativas . . . 54
4.3.8 Preposiciones . . . 55
4.3.9 Coordinaciones . . . 56
4.4 Proceso bottom-up . . . 60
4.4.1 Identificación de núcleos sintácticos . . . 62
4.4.2 Clasificación de argumentos . . . 63
4.4.3 Reglas de identificación de elementos . . . 64
4.4.4 Grupos nominales . . . 66
4.4.5 Sintagmas nominales . . . 69
4.4.6 Sintagmas y grupos adjetivales . . . 69
4.4.7 Grupos verbales . . . 71
4.4.8 Oraciones . . . 75
4.5 Evaluación . . . 78
4.5.1 Identificación de núcleos . . . 79
4.5.2 Clasificación de argumentos . . . 80

5 Extracción del léxico **83**
5.1 Entradas léxicas . . . 83
5.2 Frame léxico . . . 84
5.3 Verbos . . . 87
5.4 Nombres . . . 89
5.5 Adjetivos . . . 91
5.6 Léxico extraído . . . 91

6 Supertagging **93**
6.1 Motivación . . . 93
6.2 Antecedentes de supertagging . . . 94
6.3 Verbos . . . 95
6.3.1 Corpus . . . 95
6.3.2 Lemas . . . 97
6.3.3 Lemas y categorías gramaticales . . . 98
6.3.4 Solamente verbos con argumentos . . . 99
6.4 Nombres . . . 101
6.4.1 Corpus . . . 101
6.4.2 Experimentos . . . 102
6.5 Adjetivos . . . 103
6.5.1 Corpus . . . 104
6.5.2 Experimentos . . . 104
6.6 Supertagger unificado . . . 105

7 Discusión **107**
7.1 Conclusiones . . . 107
7.1.1 Gramática HPSG para el español . . . 107
7.1.2 Corpus de oraciones HPSG . . . 107
7.1.3 Léxico HPSG . . . 108
7.1.4 Supertagger . . . 108
7.2 Trabajo a futuro . . . 109
7.2.1 Manejo de clíticos . . . 109
7.2.2 Manejo de expresiones relativas . . . 109
7.2.3 Categorías gramaticales faltantes . . . 110
7.2.4 Mejoras en supertagging . . . 110
7.2.5 Construcción del parser . . . 110
7.2.6 Reutilización de recursos para otros formalismos . . . 111
7.2.7 Segmentación en cláusulas . . . 111

Anexos **119**

A Reglas de transformación del corpus **121**

B Evaluación de la transformación **125**

C Performance de los experimentos **127**
C.1 Verbos . . . 127
C.1.1 Experimentos con lemas . . . 127
C.1.2 Experimentos con lemas y POS . . . 128
C.1.3 Experimentos con oraciones que tienen todos los argumentos marcados . . . 130
C.2 Nombres . . . 131
C.2.1 Performance global de los experimentos . . . 131
C.2.2 Nombre genérico vs. nombre con complementos . . . 132
C.3 Adjetivos . . . 133
C.4 Supertagger unificado . . . 134

Capítulo 1

Introducción

1.1 Motivación

El proceso de análisis sintáctico o *parsing* es una actividad central dentro del procesamiento de lenguaje natural (PLN). Este proceso toma como entrada una oración y devuelve como resultado una representación de dicha oración en algún formalismo gramatical, generalmente en formato de árbol (lo que se conoce como *árbol sintáctico* de la oración). Un sistema que realiza este procesamiento se conoce como *analizador sintáctico* o *parser*. Dependiendo del formalismo que utilice, un *parser* puede además incorporar más información, no solamente sintáctica, en el resultado de su análisis.

Lo interesante de la representación en formato de árbol es que permite vincular elementos no contiguos: se pasa de una representación lineal de las palabras a una en que se puede mostrar más claramente cuáles palabras se relacionan entre sí, a pesar de que puedan estar lejos en la oración.

Se puede trazar una distinción entre lo que se conoce como *parsing* superficial (*shallow parsing* o *chunking*) y *parsing* profundo (*deep parsing*). El primero se refiere a un tipo de análisis sintáctico leve en el cual se identifican constituyentes simples: se utiliza la información morfológica de las palabras de una oración para agruparlas en unidades contiguas como grupos nominales o verbales, pero sin establecer su estructura profunda. El *parsing* profundo intenta construir una representación completa de árbol de la oración, donde cada constituyente tiene una estructura interna y a su vez se componen para formar las estructuras más grandes.

Las relaciones sintácticas que se construyen tienen un correlato semántico y por eso el análisis sintáctico nos deja un paso más cerca de poder obtener las representaciones semánticas de las oraciones. Algunos formalismos gramaticales intentan incorporar explícitamente esta información semántica dentro de sus representaciones. Por eso hablamos de formalismos con *estructuras ricas*, que sirven como frontera entre la sintaxis y la semántica.

Contar con mejores herramientas de *parsing* puede traer aparejadas mejoras

en muchas aplicaciones de PLN, como por ejemplo: inferencia textual, traducción automática, análisis de sentimiento y respuestas a preguntas. Muchas aplicaciones se apoyan sobre un análisis sintáctico para realizar su procesamiento, por lo que se beneficiarían de tener mejores *parsers* que puedan realizar el análisis más rápidamente, con mayor precisión u obteniendo estructuras más ricas con información que las aplicaciones puedan explotar mejor.

Para el idioma inglés la performance del *parsing* ha mejorado y se han construido *parsers* que devuelven análisis para formalismos gramaticales ricos con cierta efectividad. Para el español, sin embargo, son pocos los ejemplos que tenemos de *parsers* profundos que devuelvan estructuras ricas. Al inicio del presente proyecto no había *parsers* disponibles con estas características. Actualmente sí existen ejemplos, pero aún la performance de estas herramientas para el español no se acerca a la obtenida por los sistemas que trabajan con el idioma inglés.

El formalismo gramatical HPSG es un ejemplo de formalismo cuyos análisis sintácticos cuentan con estructuras ricas, donde se combina la información de los constituyentes sintácticos con otros atributos. Es habitual que las gramáticas HPSG incluyan información semántica en sus análisis. En este trabajo el objetivo es construir recursos lingüísticos para trabajar con una gramática HPSG para el español. Elegimos este formalismo debido a que existen antecedentes de *parsers* estadísticos para el idioma inglés que funcionan rápido, tienen buena cobertura y permiten análisis ricos de las oraciones del idioma. Para el español el *parser* HPSG existente está basado en una gramática con reglas construidas manualmente y su velocidad es menor. El presente trabajo plantea una alternativa a esta gramática y este *parser*, comenzando por la construcción de un corpus anotado en el formalismo HPSG del cual se puedan extraer estadísticas.

1.2 Etapas para la construcción de un parser

La construcción de un *parser* estadístico en el formalismo HPSG es un proceso largo que incluye varias etapas:

- Definición de la información que deben contener las entradas léxicas y los constituyentes de las oraciones.
- Definición de las reglas de la gramática que capturen los fenómenos lingüísticos que se desean modelar.
- Construcción de un corpus anotado utilizando las estructuras de rasgos y reglas de la gramática definida.
- Extracción del léxico del corpus: las palabras que componen el lenguaje modelado.
- Entrenamiento de un *parser* estadístico utilizando la información del corpus para estimar la frecuencia de aplicación de las reglas y de aparición de las palabras.

El alcance de la presente tesis no incluye la construcción del *parser* HPSG estadístico completo, sino la elaboración de recursos que permitan trabajar hacia la construcción de dicho *parser*.

1.3 Objetivos

El presente trabajo tiene como objetivos la generación de los siguientes recursos necesarios para la construcción de un *parser* HPSG estadístico:

- Gramática HPSG para el español: definición de las estructuras de rasgos y reglas combinatorias.
- Corpus de oraciones anotadas en la gramática definida.
- Conjunto de ítems léxicos que se utilizan en el corpus.
- *Supertagger*[1] que identifica los ítems léxicos más probables para cada palabra de una oración.

1.4 Estructura del documento

El resto del presente documento se divide en los siguientes capítulos:

El capítulo 2 presenta el problema del análisis sintáctico y algunos formalismos que se utilizan para construir representaciones sintácticas ricas. Se introducen las gramáticas HPSG y algunas herramientas destacadas (tanto para el inglés como para el español) que trabajan con este formalismo. Además se describe el corpus AnCora que se utiliza como base para la construcción del corpus HPSG en español.

El capítulo 3 presenta algunos fundamentos de las gramáticas HPSG y describe en detalle las estructuras de rasgos y las reglas de combinación de expresiones que utiliza nuestra gramática HPSG para el español.

El capítulo 4 describe el proceso de transformación del corpus AnCora de su estructura original (gramática libre de contexto enriquecida con atributos) al formato compatible con la gramática HPSG que utilizamos. Se describe además la evaluación realizada del proceso de transformación.

El capítulo 5 describe la obtención de las entradas léxicas del corpus, las cuales son agrupadas en *frames léxicos* según sus propiedades combinatorias sintáctico-semánticas.

El capítulo 6 detalla la construcción del *supertagger* para la gramática HPSG construida. Se presenta el concepto de *supertagging* y ejemplos de uso de *supertaggers* en diferentes *parsers* estadísticos. Luego se describe el entrenamiento

[1] Un *supertagger* es una herramienta que permite restringir el espacio de búsqueda del *parser* seleccionando las entradas léxicas más probables para cada palabra. Su uso es muy común en los *parsers* estadísticos de formalismos gramaticales ricos debido a la gran cantidad de entradas léxicas diferentes con las que se puede instanciar una palabra.

de los diferentes experimentos de *supertagging* realizados, enfocándose en las distintas categorías gramaticales tratadas. Finalmente se describe el *supertagger* unificado construido.

El capítulo 7 presenta las conclusiones del presente trabajo. Además se describen puntos del trabajo que quedaron sin explorar y pueden seguir profundizándose en el futuro.

Capítulo 2

Antecedentes

En este capítulo se introducen generalidades del proceso de análisis sintáctico y se describen algunos formalismos gramaticales que se utilizan para modelar dichos análisis. Se introduce además el formalismo gramatical que se utiliza en el presente trabajo, HPSG, así como algunas herramientas que utilizan este formalismo, incluyendo: SRG, una gramática HPSG para el idioma español; y Enju, un parser HPSG para el idioma inglés. Finalmente se presenta el corpus AnCora, el cual se utiliza a lo largo de este trabajo.

2.1 Formalismos gramaticales

El análisis sintáctico (o *parsing*) toma como entrada una oración en lenguaje natural y devuelve como salida una representación de la oración en un formalismo gramatical. El *parsing* puede realizarse a nivel superficial o profundo, y el resultado del análisis puede ser solamente sintáctico o además incluir otros atributos que permitan una representación más rica. En el presente trabajo nos interesarán particularmente los formalismos gramaticales que representan estructuras ricas.

La salida de un *parser* siempre es una representación de la oración en algún formalismo gramatical, y existen muchos formalismos con diferentes propiedades y poder expresivo.

Uno de los primeros formalismos en utilizarse es el de las Gramáticas Libres de Contexto (GLC o en inglés CFG: *Context Free Grammars*). Un árbol de *parsing* de una GLC es un árbol de constituyentes, conteniendo información solamente sintáctica. Para el idioma inglés, un *parser* estadístico (de tipo GLC probabilística) alcanza 88,1% de *precision*[1] y 87,5% de *recall*[2] [1]. En [2] se

[1] *Labeled Precision*: cantidad de constituyentes encontrados y etiquetados correctamente sobre el total de constituyentes encontrados.

[2] *Labeled Recall*: cantidad de constituyentes encontrados y etiquetados correctamente sobre el total de constituyentes correctos.

reporta que este tipo de *parsers* pueden alcanzar una performance de alrededor de 92% de medida-F_1 para el idioma inglés.

Si bien las GLC permiten modelar muchas construcciones del lenguaje, en muchos casos resultan poco expresivas para modelar fenómenos lingüísticos comunes. Por ejemplo, para modelar la concordancia entre palabras (en español podría ser concordancia sujeto-verbo o nombre-adjetivo) es necesario crear múltiples no terminales y reglas que especifiquen las distintas variantes de las categorías gramaticales, lo cual llevaría a una explosión combinatoria de reglas. Algo similar ocurre al intentar modelar los verbos de control sujeto o control objeto. Estos problemas se pueden resolver fácilmente utilizando otros tipos de gramáticas que trabajen con estructuras de rasgos.

En general los métodos que tienen mayor performance pertenecen a la familia de los formalismos gramaticales lexicalizados. En las gramáticas lexicalizadas, las palabras (o entradas léxicas) incorporan la información acerca de las restricciones combinatorias que tienen respecto a otras palabras. Dicha información guía el proceso de *parsing*. En los formalismos lexicalizados se necesita definir un elemento distinguido en cada subárbol de *parsing* que actúa como ancla para combinar el subárbol con otros subárboles dentro del análisis. A este elemento se le suele denominar núcleo léxico o *head*.

Un ejemplo son las Gramáticas de Dependencias (en inglés: *Dependency Grammars*), que no se basan en la identificación de constituyentes sino que establecen relaciones (dependencias) entre las palabras de la oración. Cada palabra tiene una dependencia hacia otra palabra (denominada núcleo), dicha dependencia se nota como un arco y lleva una etiqueta que indica el tipo de relación (por ejemplo: sujeto, objeto directo, especificador). Si bien se suele hablar de gramáticas de dependencias, estos tipos de análisis no incluyen estrictamente una gramática formal debido a que no definen reglas y restricciones para la composición de constituyentes como lo hacen las otras gramáticas, en cambio las relaciones se establecen directamente entre pares de palabras. En lo que refiere a la performance de estos métodos de análisis podemos citar como ejemplo el *parser* de dependencias de Stanford, que utiliza redes neuronales para obtener la representación en *features* de las palabras y alcanza un UAS[3] de 92,0% y un LAS[4] de 90,7% [3].

Las Gramáticas Combinatorias Categoriales (en inglés CCG: *Combinatory Categorial Grammars*) [4] son gramáticas que construyen categorías sintácticas complejas a partir de otras categorías simples utilizando los operadores de aplicación a la derecha y a la izquierda. Por ejemplo: un determinante puede verse como un elemento que se aplica a la izquierda de un nombre (categoría N) y devuelve un NP, esto se nota como $N\backslash NP$. Un *parser* estadístico para este tipo de gramáticas alcanza un 81,6% de *precision* y un 81,9% de *recall* [5].

Las Gramáticas de Adjunción de Árboles (en inglés TAG: *Tree Adjoining Grammars*) [6] son gramáticas de sustitución de árboles. Las estructuras básicas

[3]Unlabeled Attachment Score: porcentaje de los palabras para las que se identificó correctamente el núcleo.

[4]Labeled Attachment Score: porcentaje de las palabras para las que se identificó correctamente el núcleo y además la etiqueta del arco.

de la gramática son árboles de dos tipos: los árboles iniciales representan estructuras simples y los árboles auxiliares permiten recursión sobre las estructuras. Estos árboles se combinan mediante reglas de sustitución y de adjunción [7]. Un *parser* estadístico para este tipo de gramáticas alcanza un 86,9% de *recall* y un 86,6% de *precision* en oraciones de hasta 40 palabras [8]. Desde el punto de vista formal, tanto las TAG como las CCG son gramáticas de tipo *mildly context sensitive* [9], lo cual significa que son más expresivas que las GLC y permiten modelar un conjunto más amplio de lenguajes, pero mantienen la característica de tener algoritmos de *parsing* que pueden reconocer una oración en tiempo polinómico [10].

Para el idioma español se destaca el *parser* de dependencias EsTxala [11], utilizado por la suite de herramientas de análisis lingüístico FreeLing [12]. EsTxala es un *parser* de dependencias basado en reglas escritas manualmente, que alcanza un UAS de 81,13% y un LAS de 73,88%. También existe una versión del *parser* de dependencias MaltParser [13] entrenada para el español que reporta un LAS de 81.29% [14].

2.2 HPSG

En este trabajo nos concentramos en un formalismo gramatical denominado Gramáticas Sintagmáticas Nucleares (en inglés HPSG: *Head-Driven Phrase Structure Grammars*) [15]. Son gramáticas fuertemente lexicalizadas donde las palabras y los árboles son estructuras de rasgos tipificadas (en inglés TFS: *typed feature structures*) [16] y se encuentran organizadas en una jerarquía de tipos. Cada tipo tiene rasgos (*features*) y define valores posibles para esos rasgos. Las entradas léxicas se combinan con las reglas utilizando operaciones de *unificación* sobre las estructuras de rasgos.

El hecho de que la mayor parte de la información combinatoria de la gramática esté incluida en las entradas léxicas implica que este tipo de gramáticas contienen relativamente pocas reglas para la construcción de las frases (comparadas por ejemplo con las Gramáticas Libres de Contexto). Estas reglas son genéricas, inspiradas en la teoría X' ([17], [18]), e incluyen reglas para juntar un núcleo con su especificador, sus complementos o sus modificadores.

En cuanto a la expresividad para modelar construcciones lingüísticas, debido a que HPSG es una gramática basada en estructuras de rasgos no tiene los problemas mencionados para las GLC. Desde un punto de vista de los lenguajes formales, en [19] se indica que las gramáticas HPSG con reglas léxicas simples se pueden utilizar para describir lenguajes recursivamente enumerables, lo cual las hace más expresivas que las GLC, que describen lenguajes libres de contexto y también más expresivas que las gramáticas tipo CCG y TAG que son *mildly context sensitive*.

2.2.1 DELPH-IN

Si bien la gramática HPSG original surgió como descripción del idioma inglés, la teoría es lo suficientemente genérica y flexible como para modelar las gramáticas de muchos idiomas. En particular DELPH-IN [20] es una iniciativa para crear gramáticas HPSG para diferentes idiomas. Las gramáticas creadas comparten ciertas características que les permiten algún nivel de interoperabilidad, lo cual puede utilizarse por ejemplo para soportar la traducción automática basada en semántica entre los idiomas participantes [21].

Uno de los proyectos asociados a DELPH-IN es la *LinGO Grammar Matrix* [22], un *framework* para crear gramáticas HPSG que ha sido utilizado para crear gramáticas para el noruego, coreano, portugués y español, entre otros. Como parte de DELPH-IN también está el sistema *Linguistic Knowledge Builder* (LKB) [23], un sistema para implementar gramáticas basadas en restricciones, muy usado en el desarrollo de gramáticas HPSG. El proceso de *parsing* es basado en reglas (es un *chart parser*). Además de LKB, DELPH-IN provee otro sistema para trabajar con gramáticas de unificación denominado PET [24], que utiliza el mismo lenguaje que LKB e incluye un mecanismo de clasificación (*ranking*) de los resultados de *parsing* de acuerdo a modelos estadísticos de selección.

2.2.2 Spanish Resource Grammar

La gramática para el español Spanish Resource Grammar (SRG) [25] fue desarrollada partiendo del *framework* LinGO. Es la gramática HPSG más completa para el español desarrollada hasta el momento. SRG contiene más de 50.000 entradas léxicas, 64 reglas léxicas y 191 reglas de combinatoria para formar frases. Las reglas fueron desarrolladas manualmente y se utilizaron estrategias manuales y automáticas para construir el léxico [26]. Utiliza el enfoque semántico de *Minimum Recursion Semantics* (MRS) [27] que es el modelo semántico sugerido por DELPH-IN. Los resultados del análisis de oraciones utilizando la SRG son árboles muy ricos que soportan muchas de las construcciones previstas por la teoría.

Debido a que utiliza el *parser* LKB, en su versión original esta gramática no incluye soporte para un análisis estadístico de los árboles de *parsing*, por ejemplo para identificar cuál de los posibles análisis de una misma oración es el más probable. Más adelante se incorpora un método de desambiguación de los posibles análisis [28] para permitir la construcción de un corpus de árboles anotados, que se basa en la siguiente idea:

- Obtener todos los análisis de las oraciones de un corpus utilizando el *parser*.
- Elegir manualmente el mejor de los análisis para cada oración.
- Entrenar un modelo de entropía máxima y otro basado en *parsing* de dependencias que intente predecir el mejor análisis de cada oración.

- Si los dos modelos predicen el mismo análisis como el más probable, se elige ese análisis, de lo contrario se lo envía a desambiguar manualmente por un humano.

De esta manera se puede iterar e ir construyendo un corpus más grande y entrenar mejores predictores. Una vez que se construyó el corpus se utilizaron reglas automáticas para convertirlo a un formato de gramáticas de dependencias. El resultado de este proceso es el corpus IULA Spanish LSP TreeBank [29], un corpus de 40.000 oraciones en español. Este corpus está etiquetado con información de dependencias, pero el corpus original con los análisis en formato HPSG desambiguados para cada oración que se utilizó para construir el de dependencias no está disponible.

El enfoque descrito da buenos resultados. A medida que aumenta la cantidad de oraciones etiquetadas permitiría construir mejores clasificadores para desambiguar el análisis. Sin embargo, tiene el inconveniente de que aún se debe utilizar el *chart parser* para obtener todos los posibles análisis y luego realizar un post-procesamiento, lo cual es en general más lento que utilizar las probabilidades para guiar el *parsing* y descartar los análisis poco prometedores.

En el presente trabajo se propone un enfoque alternativo para la construcción de una gramática HPSG para el español: partiendo de un corpus se construye la gramática y las entradas léxicas junto con un conjunto de ejemplos ya etiquetados, lo cual permitiría obtener probabilidades de las entradas léxicas y las reglas. Sin embargo, se debe tener en cuenta que la riqueza de las estructuras que se pueden modelar depende de la información original contenida en el corpus.

2.2.2.1 Ejemplos

Se realizaron experimentos de *parsing* de algunas oraciones[5] utilizando la demostración en línea del *parser* SRG[6], que permite devolver los árboles analizados ordenados por prioridad. Las oraciones que se utilizaron para estos experimentos son las siguientes:

1. El número de desplazados por las inundaciones ascendió a 572 personas en Artigas, Cerro Largo, Paysandú, Rivera y Salto.

2. Dos juezas de Treinta y Tres son investigadas por la SCJ por supuestas irregularidades y por protagonizar episodios de abuso de poder.

3. Un ómnibus que llevaba cerca de 20 uruguayos con destino a Salta, se accidentó en Tucumán.

Ninguna de las tres oraciones completas pudo ser analizada por el *parser*, debido a que la demostración en línea tiene una limitación en el tamaño del *chart* que se permite para el *parsing*. Por lo tanto, se probaron versiones simplificadas de las mismas que sí fuera posible analizar.

[5] Titulares de prensa obtenidos del portal de noticias Montevideo COMM (`http://www.montevideo.com.uy/`) el 18/10/2015.

[6] `http://logon.iula.upf.edu/logon`

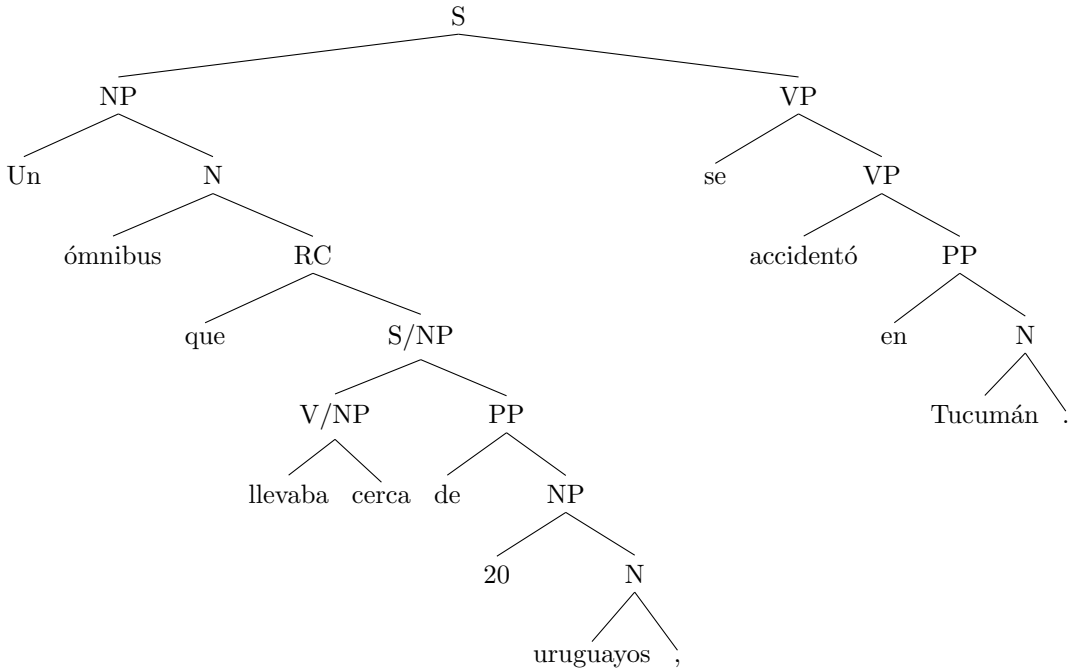

Figura 2.1: Esqueleto del análisis SRG para "Un ómnibus que llevaba cerca de 20 uruguayos, se accidentó en Tucumán."

La figura 2.1 muestra el esqueleto del análisis de "Un ómnibus que llevaba cerca de 20 uruguayos, se accidentó en Tucumán.". Por simplicidad, se muestra el árbol de constituyentes resultante del análisis sin los rasgos de cada nodo. El proceso de *parsing* tomó 17,68 segundos y el resultado es en su mayoría correcto. Por ejemplo se identificó correctamente que el sujeto del verbo de la oración relativa "llevaba" es el ómnibus. El único error es que el adverbio "cerca" no quedó unido a su complemento "de 20 uruguayos", lo que dejó un análisis equivocado de dicho subárbol.

La figura 2.2 muestra el análisis de "El número de desplazados por las inundaciones ascendió a 572 personas en Artigas y Salto.". El proceso de *parsing* tomó 17,99 segundos y el único error que tiene ocurre al analizar la coordinación: el análisis correcto debería coordinar "Artigas y Salto", y esta frase coordinada sería el complemento de la preposición "en".

La figura 2.3 muestra el análisis de la oración "Dos juezas son investigadas por la SCJ por supuestas irregularidades y por protagonizar episodios de abuso de poder.". El proceso de *parsing* tomó 17,50 segundos. En este caso el *parser* no fue capaz de determinar correctamente los argumentos del verbo "investigadas". Se detecta que el sujeto es "dos juezas" pero se indica que el complemento es un solo grupo preposicional que incluye todo el resto de la oración. El grupo preposicional coordinado "por supuestas irregularidades y por protagoni-

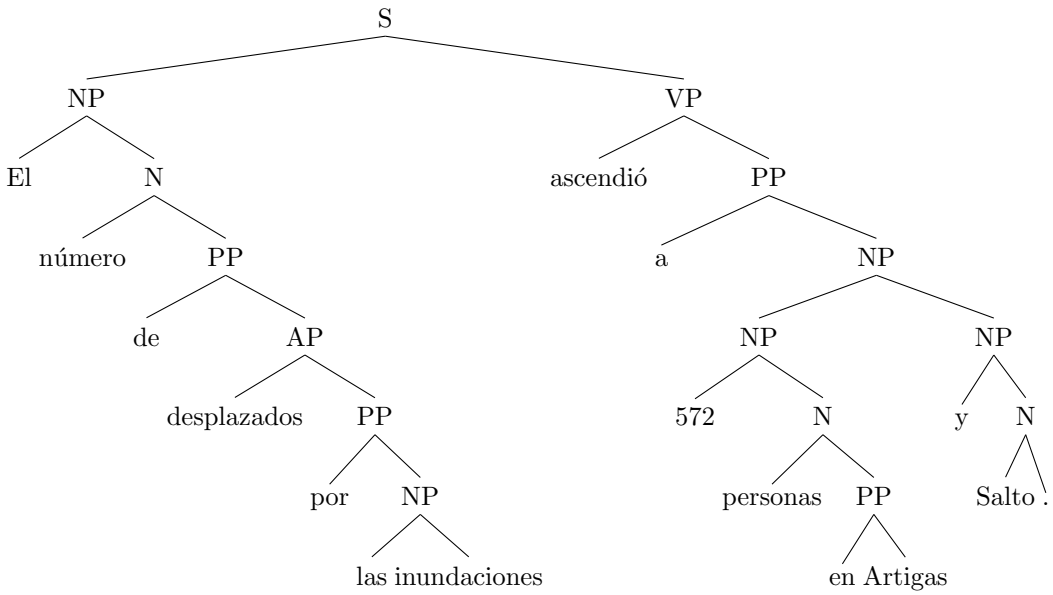

Figura 2.2: Esqueleto del análisis SRG para "El número de desplazados por las inundaciones ascendió a 572 personas en Artigas y Salto."

zar episodios de abuso de poder" queda bien analizado, pero no se detecta que este es un complemento del verbo.

Una posibilidad es que el nombre "SCJ" (Suprema Corte de Justicia) esté confundiendo al *parser* y se esté asumiendo que el otro grupo preposicional depende de él. Por lo tanto se realizó un experimento simplificando más la oración. La figura 2.4 muestra el análisis de la oración "Dos juezas son investigadas por irregularidades y por abuso de poder.". El proceso de *parsing* tomó 2,77 segundos, el ejemplo más rápido de los que se probaron. Sin embargo, el análisis en este caso presentó otros errores, ya que se detectó que el único complemento de "investigadas" es "por irregularidades". La frase "y por abuso de poder" aparece como coordinada con la anterior, y ambas aparecen como argumento de "son", lo cual es un error.

Notamos que el proceso de *parsing* puede ser lento para oraciones largas (alrededor de 17 segundos), lo que puede deberse a que se estarían obteniendo todos los posibles análisis y calculando la probabilidad de cada uno para devolver el más relevante. Utilizando el *parser* sin restricciones en el espacio de búsqueda probablemente puedan analizarse ejemplos más complejos, pero es posible que tanto el tiempo de análisis como la cantidad de errores en el análisis aumenten considerablemente.

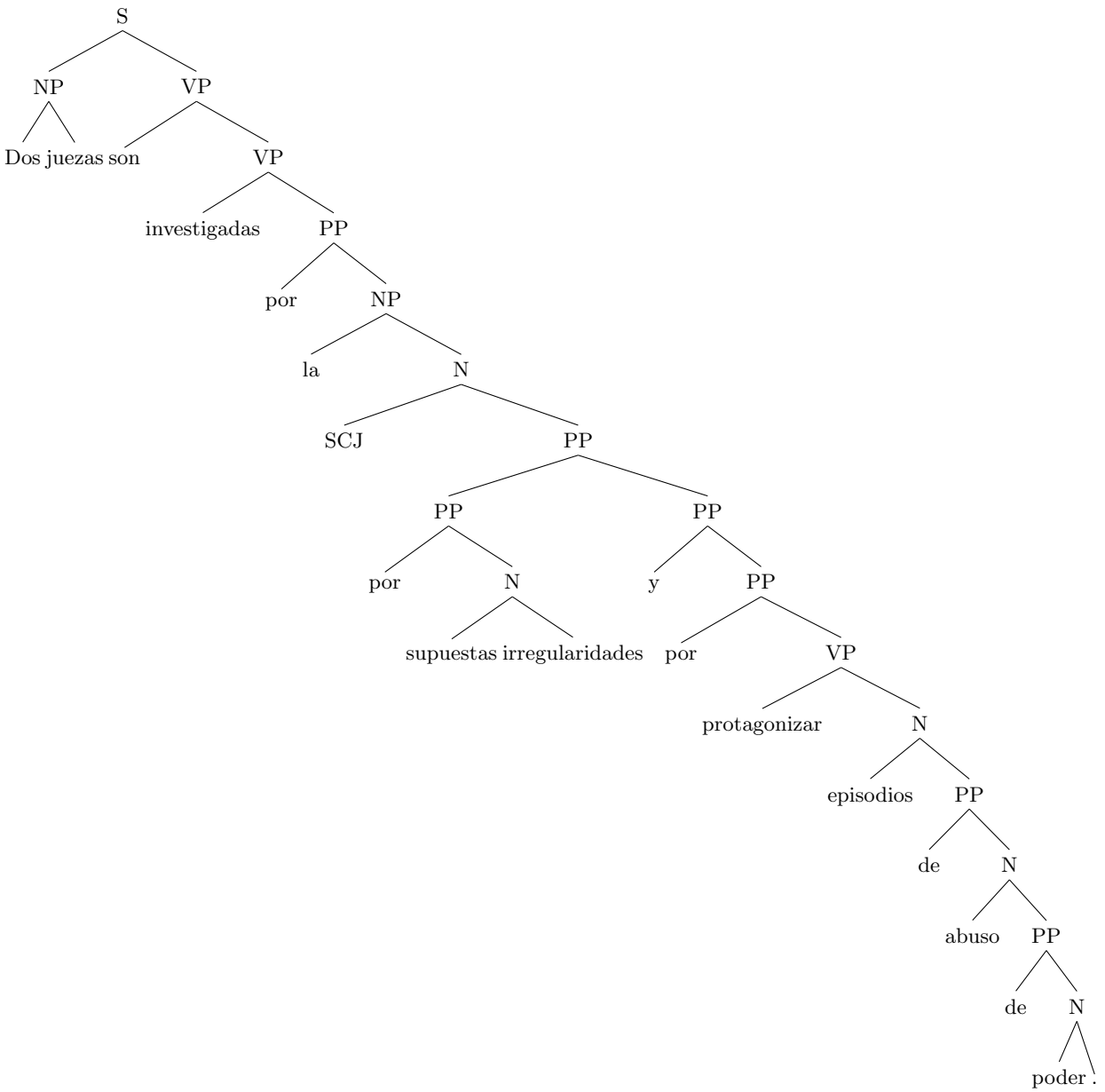

Figura 2.3: Esqueleto del análisis SRG para "Dos juezas son investigadas por la SCJ por supuestas irregularidades y por protagonizar episodios de abuso de poder."

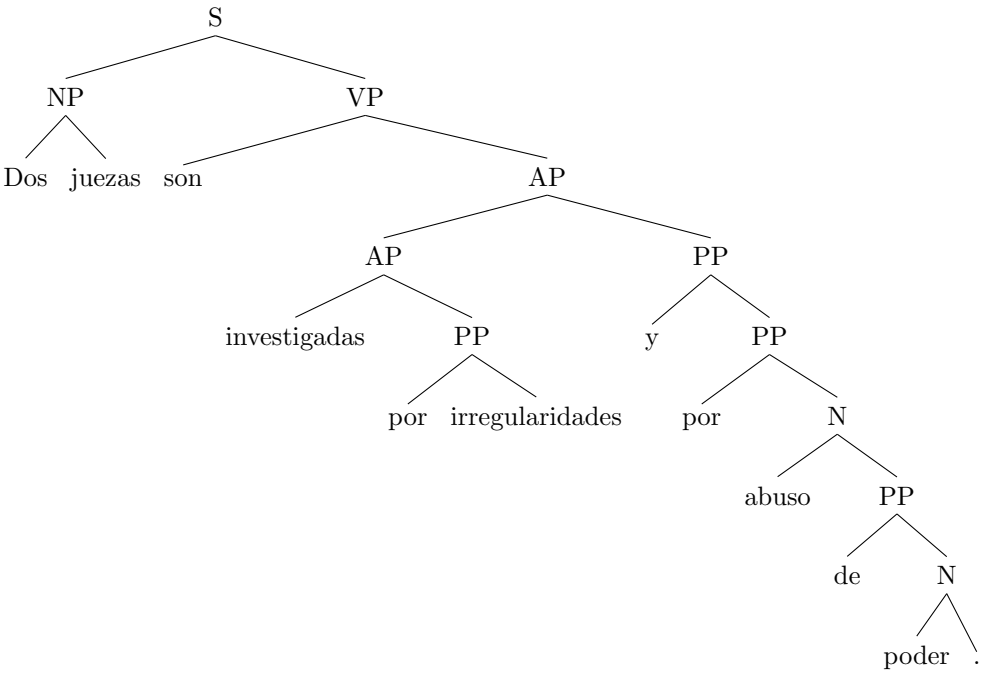

Figura 2.4: Esqueleto del análisis SRG para "Dos juezas son investigadas por irregularidades y por abuso de poder."

2.2.3 Enju

La estrategia que seguimos está inspirada en la del sistema Enju [30], un *parser* HPSG estadístico de alta cobertura para el idioma inglés. Enju es un *parser* de alta performance y cobertura del idioma que además resuelve con éxito algunos fenómenos lingüísticos complejos como los verbos de alzamiento y de control, y también es capaz de analizar con cierto éxito los sintagmas coordinados.

Para la construcción de este *parser* se tomó como base el corpus Penn Treebank [31], un corpus para el idioma inglés de 4,5 millones de palabras. Este corpus es muy utilizado para el desarrollo de *parsers* estadísticos debido a que contiene una sección de un millón de palabras cuyas oraciones están etiquetadas sintácticamente utilizando una gramática libre de contexto enriquecida con atributos. Como las anotaciones sintácticas de este corpus no son directamente compatibles con una gramática HPSG, se implementó un proceso de transformación que adapta las reglas del Penn Treebank a un formato similar a HPSG [32]. Luego se utilizó el corpus resultante para construir las entradas léxicas y extraer las probabilidades de aplicación de las reglas.

Enju provee dos posibles *parsers*:

- El *parser* Enju que primero utiliza un *supertagger* para obtener las entradas léxicas más probables para las palabras de la oración y aplica el algoritmo CKY estadístico sobre esas posibilidades para obtener el *parsing* [33]. Este proceso devuelve análisis con muy alta precisión, aunque puede ser un poco más lento (500 ms por oración).

- El *parser* Mogura que utiliza el *supertagger* pero solamente se queda con una entrada léxica para cada palabra y luego utiliza un algoritmo *shift-reduce* para realizar el análisis [34]. Este proceso es mucho más rápido (alrededor de 50 ms por oración) a expensas de que en algunos casos el análisis no será el mejor posible.

Central a ambos procesos de *parsing* es el concepto de *supertagger*. Un *supertagger* es un sistema que elige las entradas léxicas más probables para cada palabra de una oración, como una especie de generalización del *tagger* que solamente elige la categoría gramatical de las palabras. Inicialmente propuesto en [35] para su uso en una Lexicalized Tree Adjoining Grammar, el *supertagging* se ha vuelto una etapa común en el proceso de *parsing* para todas las gramáticas lexicalizadas, debido a que en este tipo de gramáticas cada palabra puede tener numerosas categorías posibles. El *supertagging* limita la explosión combinatoria de posibilidades para las palabras, lo que permite que la búsqueda de los árboles de *parsing* más probables sea un problema tratable.

El *supertagger* implementado para Enju [36] utiliza una gramática libre de contexto que aproxima la gramática HPSG utilizada por Enju. Esta GLC permite descartar rápidamente análisis que no son posibles, y este dato es incorporado al proceso de entrenamiento del *supertagger*, de manera de aumentar su precisión. Con esto logran un nivel de exactitud del 93.98%.

A diferencia de las gramáticas soportadas por DELPH-IN, Enju no utiliza la plataforma LKB para la manipulación de las estructuras de rasgos, sino que tienen un lenguaje propio denominado LiLFeS [37]. LiLFeS es un lenguaje que se basa en el paradigma de la programación lógica y su tipo de dato fundamental es la estructura de rasgos tipificada.

Los rasgos semánticos de la gramática utilizada por Enju también difieren del enfoque semántico propuesto por DELPH-IN: en vez de modelar la semántica mediante MRS se limitan a incluir rasgos para la estructura argumental semántica de los verbos de la oración. Esto es posible gracias al corpus PropBank [38] que sistematiza las anotaciones de la estructura argumental de las oraciones del Penn Treebank.

2.2.3.1 Ejemplos

Se realizaron experimentos de *parsing* de algunas oraciones[7] utilizando la demostración en línea del *parser* Enju[8], que devuelve el análisis más probable para una oración. Las oraciones que se utilizaron para estos experimentos son las siguientes:

1. The smallest falcon in New York City, the American kestrel is a fast and deadly foe to smaller birds, snakes and mice.

2. Two new plays by female writers that explore both the virtual and real worlds have premieres in the Garden State this fall.

3. Bruce G. Bukiet, who forecasts baseball standings, had his worst year in 2015.

Las tres oraciones pudieron ser analizadas por el *parser*. Como la demostración en línea no provee información acerca del tiempo que tomó el *parser* en realizar el análisis, se tomó ese tiempo manualmente, por lo que los tiempos incluyen no solo el análisis del *parser* si no también el tiempo de procesamiento y respuesta del servidor web.

El análisis de la oración (1) se muestra en la figura 2.5. El sitio tardó 2.01 segundos en mostrar el resultado. Se muestra un análisis correcto para la coordinación incluida en "fast and deadly foe" y la coordinación triple "smaller birds, snakes and mice". Sin embargo, hay un error al asociar "the American kestrel" como inciso de "New York City", cuando lo correcto es asociarlo a la frase entera "the smallest falcon in New York City".

El análisis de la oración (2) se muestra en la figura 2.6. El sitio tardó 1.81 segundos en mostrar el resultado. El análisis para la oración es correcto, incluyendo la coordinación contenida en "the virtual and real worlds" y la relativa introducida por "that", en la que se identifica correctamente que el sujeto de "explore" es "two new plays by female writers".

[7] Titulares de prensa obtenidos del portal de noticias The New York Times (`http://www.nytimes.com/`) el 18/10/2015.

[8] `http://www.nactem.ac.uk/enju/demo.html`

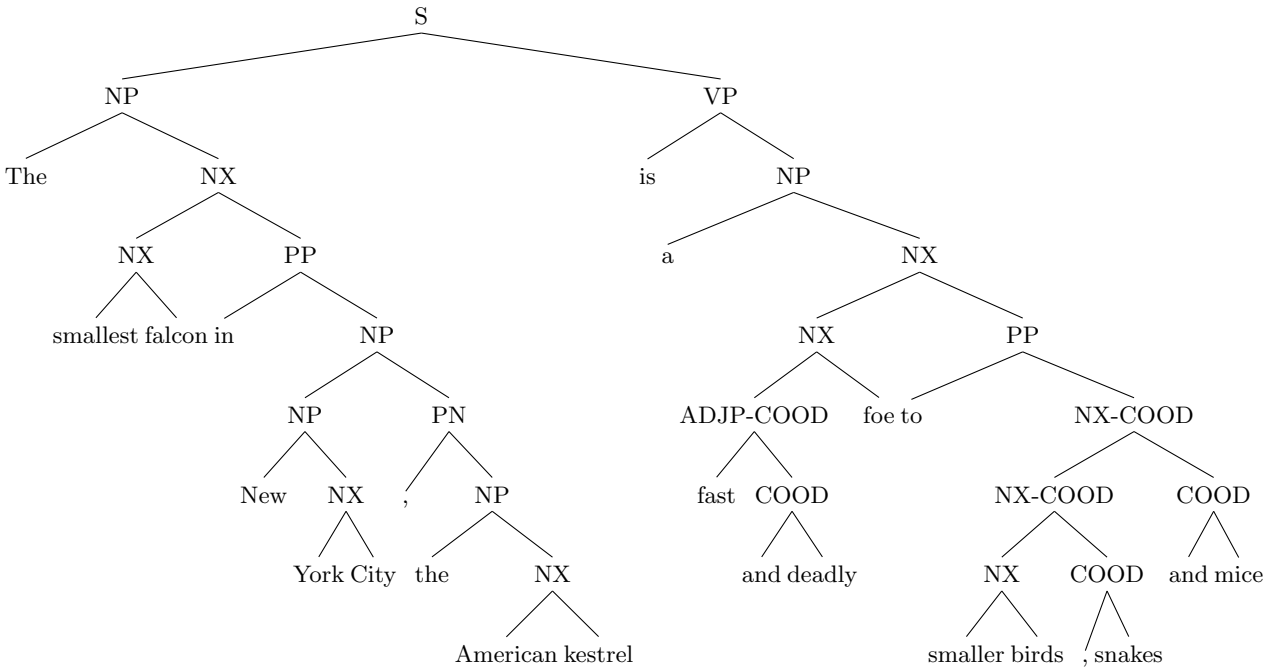

Figura 2.5: Esqueleto del análisis Enju para “The smallest falcon in New York City, the American kestrel is a fast and deadly foe to smaller birds, snakes and mice.”

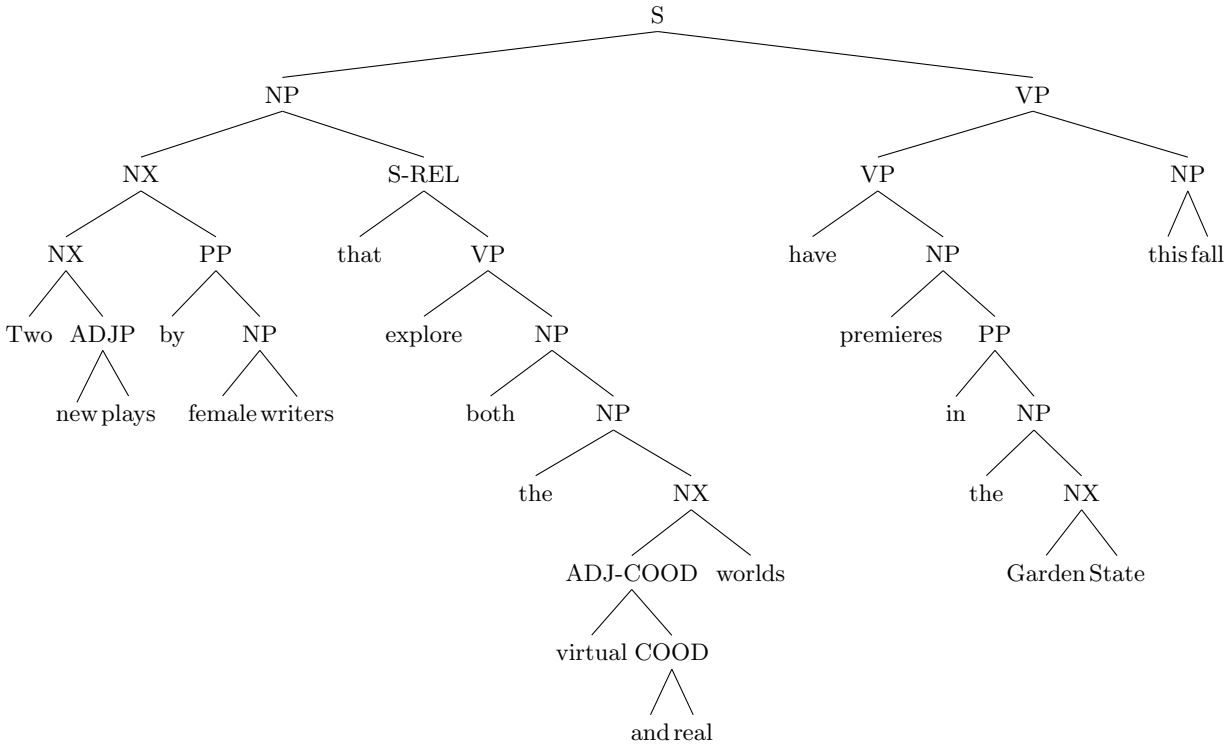

Figura 2.6: Esqueleto del análisis Enju para "Two new plays by female writers that explore both the virtual and real worlds have premieres in the Garden State this fall."

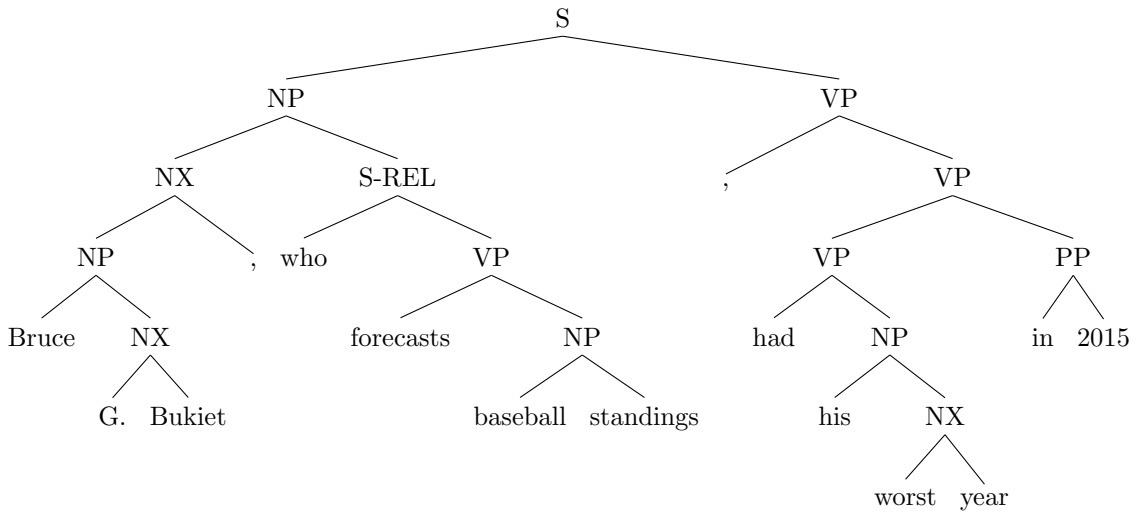

Figura 2.7: Esqueleto del análisis Enju para "Bruce G. Bukiet, who forecasts baseball standings, had his worst year in 2015."

El análisis de la oración (3) se muestra en la figura 2.7. El sitio tardó 1.59 segundos en mostrar el resultado. El análisis para la oración es correcto. Por ejemplo se identificó que el sujeto del verbo "forecasts" en la relativa es "Bruce G. Bukiet".

En general el desempeño del *parser* es bueno y responde rápidamente a las consultas a pesar de que las oraciones puedan ser largas, aunque en estos casos la probabilidad de encontrar errores en el análisis es mayor.

2.3 AnCora

El corpus AnCora [39] es un corpus de textos en español y catalán de aproximadamente 500.000 palabras. Contiene 17.000 oraciones repartidas en alrededor de 1.600 artículos, principalmente de prensa. Todas las oraciones están etiquetadas sintácticamente utilizando un formalismo similar al de las gramáticas libres de contexto. Además, en las palabras y los constituyentes se incorpora más información en forma de atributos, por ejemplo:

- Atributos morfológicos de las palabras.
- Función gramatical de los constituyentes.
- Estructura argumental.
- Sentidos de WordNet para los sustantivos.

Para la construcción de AnCora se utilizó un proceso automático de etiquetado morfológico y *parsing* superficial y a partir de estos datos se realizó un etiquetado manual de la estructura sintáctica profunda de las oraciones [40].

Además del corpus de texto etiquetado, se construyeron los recursos léxicos AnCora-Verb (que contiene 2.647 verbos para el español con sus posibles subcategorizaciones) y AnCora-Nom (con 1.600 nombres deverbales para el español con sus posibles subcategorizaciones).

Si bien las anotaciones sintácticas de AnCora siguen una estructura similar a una gramática libre de contexto, es posible transformarlas a un formato compatible con HPSG siguiendo un proceso similar al utilizado por Enju. Además, entre los atributos que se anotan para los constituyentes está la información sobre la estructura argumental de los verbos en un formato análogo al de PropBank. Esto permite que se puedan agregar rasgos para modelar la estructura argumental a la estructura de rasgos de nuestra gramática. Debido a esto se decidió utilizar AnCora como base en el presente trabajo.

Capítulo 3

Gramática propuesta

En este capítulo se describe la gramática HPSG utilizada en el presente trabajo. Comienza con una breve introducción a las estructuras de rasgos tipificadas que se utilizan para construir gramáticas HPSG. Luego se describen las estructuras de rasgos que se utilizan para modelar las palabras y las frases de nuestra gramática, así como las reglas que utilizaremos para combinar expresiones. La mayoría de las reglas que utilizamos son estándares de la teoría, pero introducimos algunas reglas como simplificación para el tratamiento de ciertos casos complejos.

3.1 Fundamentos

En una gramática HPSG, todas las estructuras (palabras, frases, reglas y el árbol mismo de *parsing*) se representan mediante estructuras de rasgos tipificadas (TFS). Una TFS es un grafo dirigido con etiquetas en los nodos y las aristas. Las etiquetas de los nodos corresponden a *tipos*, mientras que las de las aristas corresponden a *rasgos* o *atributos* (en inglés *features*). Una TFS tiene una representación como grafo, pero puede visualizarse de manera sencilla como una matriz atributo-valor. Por ejemplo, la figura 3.1 muestra una TFS simple y su correspondiente matriz atributo-valor.

Dos TFS pueden combinarse mediante la operación de unificación. El resultado es una nueva TFS que contiene los pares <rasgo,valor> de las dos estructuras al mismo tiempo. A su vez los valores son unificados, por lo que la operación de unificación se aplica recursivamente. La figura 3.2 muestra un ejemplo de unificación de dos TFS. Notar que el resultado contiene todos los rasgos de las dos estructuras originales, y los valores correspondientes de cada rasgo también están unificados.

Puede ocurrir que en algunos casos la unificación no sea posible, debido a que los tipos de las estructuras (o los tipos de los valores asociados a alguno de sus rasgos) no sean compatibles. En ese caso la operación de unificación falla.

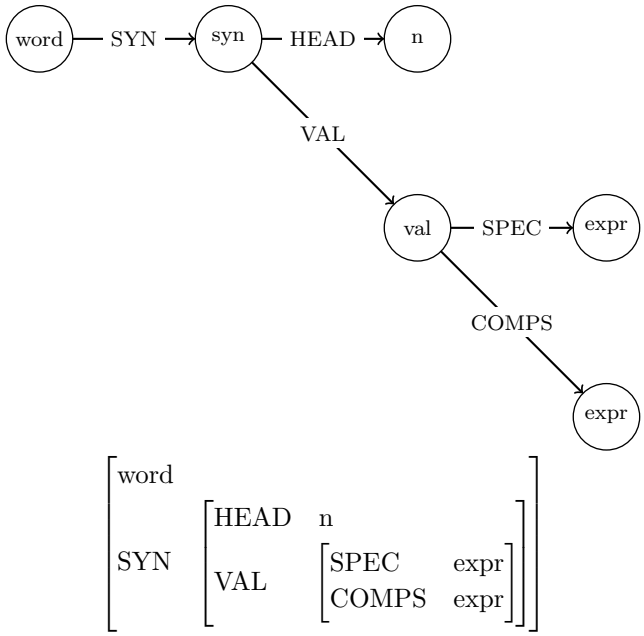

Figura 3.1: TFS simple en formato grafo y en formato matriz atributo-valor

$$\begin{bmatrix} \text{word} & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \text{n} \\ \text{VAL} & \begin{bmatrix} \text{SPEC} & \text{expr} \\ \text{COMPS} & \text{expr} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{expr} & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{n} & \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{GEN} & \text{f} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{word} & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{n} & \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{GEN} & \text{f} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{SPEC} & \text{expr} \\ \text{COMPS} & \text{expr} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figura 3.2: Ejemplo de unificación de dos TFS

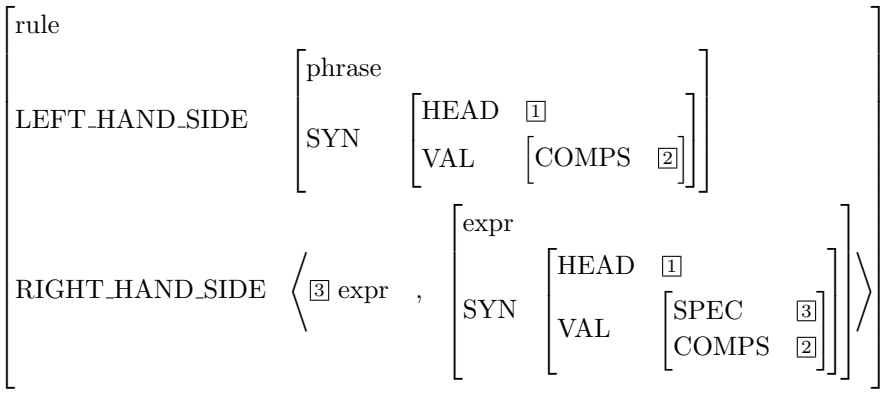

Figura 3.3: TFS para la regla `spec_head` sin instanciar

Las gramáticas HPSG contienen en general pocas reglas que combinan expresiones para formar expresiones nuevas. Estas reglas también se representan mediante TFS. Una TFS para una regla es una estructura que tiene los siguientes rasgos:

- `right_hand_side`: El lado derecho de la regla es una lista de expresiones a combinar. Una de estas expresiones actúa como núcleo (*head*) de la regla.
- `left_hand_side`: El lado izquierdo de la regla es la expresión resultante de aplicar la regla a todas las expresiones que aparecen del lado derecho.

La expresión del lado izquierdo de una regla tiene valores coindizados con las expresiones del lado derecho. De esta manera, instanciar una regla (aplicar la regla) implica unificar las expresiones del lado derecho, y el resultado de la unificación dejará del lado izquierdo una nueva expresión que será la frase resultante de aplicar la regla.

Por ejemplo, considerar la siguiente regla del especificador:

- `spec_head` → `spec head`

Esto es una notación de conveniencia para especificar que la estructura de rasgos de la regla es la que se muestra en la figura 3.3. El lado derecho indica que el rasgo `SYN.VAL.SPEC` del núcleo y la expresión que actúa como especificador están coindizados, lo cual implica que estas expresiones deben unificar. El resultado de la aplicación de la regla construye una nueva expresión que se queda con el mismo rasgo `SYN.HEAD` que el núcleo, y el rasgo `SYN.VAL.SPEC` pasa a estar saturado.

Para visualizar de manera más sencilla las aplicaciones de reglas, es habitual representarlas mediante árboles en vez del uso de las matrices atributo-valor. La regla anterior se representa como lo muestra la figura 3.4.

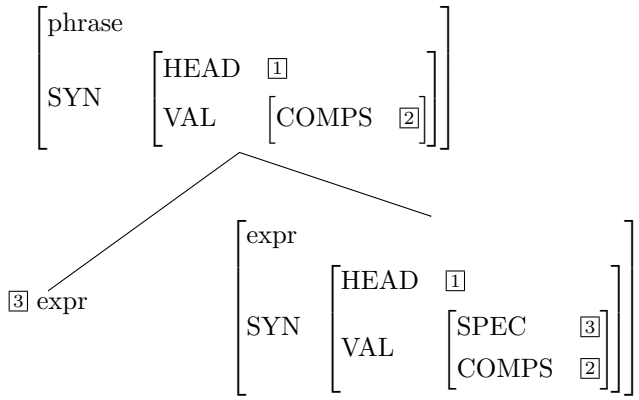

Figura 3.4: Árbol para la regla `spec_head` sin instanciar

El árbol de *parsing* de una gramática HPSG surge de la aplicación consecutiva de las reglas. Cada aplicación de una regla genera una nueva frase que se podrá utilizar del lado derecho de otra regla. El resultado es un árbol cuya estructura queda codificada en los rasgos unificados de las reglas instanciadas.

3.2 Estructuras de rasgos

El tipo central a tener en cuenta para nuestra gramática es el de las expresiones (`expr`): un árbol sintáctico es un árbol donde cada nodo es una expresión. Las expresiones se dividen en dos subtipos: palabras (`word`, que serán las hojas del árbol) y frases (`phrase`, que serán los nodos no terminales). Estas estructuras de rasgos son muy parecidas a las propuestas por la teoría HPSG [41], con algunas modificaciones que se mencionarán.

3.2.1 Palabra

La estructura de rasgos (o signo) para una palabra (tipo `word`) se muestra en la figura 3.5. A la estructura de rasgos correspondiente a una palabra se le denomina también *entrada léxica*. Esta estructura de rasgos contiene información morfológica, sintáctica y semántica acerca de la palabra. El rasgo `SYN` agrupa todos los rasgos relacionados con la combinatoria sintáctica y el rasgo `SEM` contiene lo relacionado con la semántica.

Dentro de `SYN`, en primer lugar la palabra contiene un rasgo `HEAD` cuyo valor es la categoría gramatical. Además este rasgo incluye la información morfológica en el subrasgo `ARG`, que se utiliza para aplicar restricciones de concordancia en ciertos contextos. Los rasgos morfológicos que se modelan dependen de la categoría gramatical.

El rasgo `VAL` contiene la información de combinatoria sintáctica de la palabra. Dentro de este rasgo se pueden ver los subrasgos `SPEC` y `COMPS` que indican el (posible) especificador de la palabra y la lista de complementos que puede

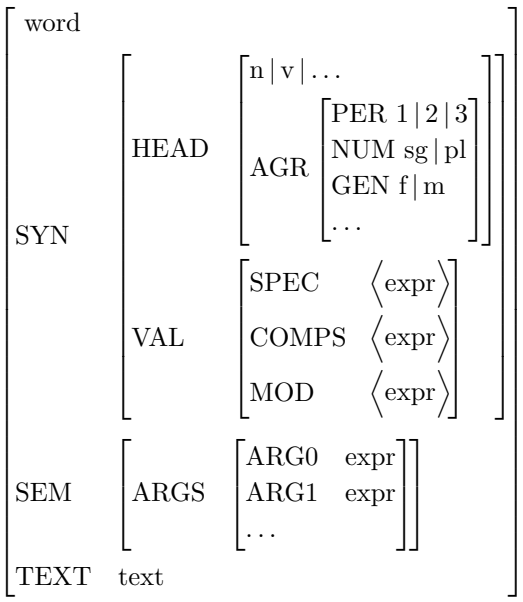

Figura 3.5: Estructura de rasgos para una palabra

tener. También se encuentra el subrasgo `MOD` que indica que esta palabra puede utilizarse como modificador de otra expresión.

Nuestro enfoque en cuanto a semántica se encuentra modelado mediante el rasgo `SEM`. Lo que se modela como información semántica es la estructura argumental de las palabras, utilizando rasgos similares a los atributos de PropBank[42], por ejemplo: el rasgo `ARG0` apunta a la expresión que representa el agente, el rasgo `ARG1` apunta a la expresión que representa el tema, etc.

En la estructura de rasgos de una palabra los atributos de valencia sintáctica y los semánticos están coindizados. Los rasgos sintácticos que se tienen en cuenta son `SPEC` y `COMPS`, ya que `MOD` no aporta información sobre estructura argumental. Diferentes coindizaciones permiten modelar fenómenos que existen en español como la voz pasiva. Consideremos las siguientes oraciones:

1. “el niño come la manzana”

2. “la manzana es comida por el niño”

Las entradas léxicas correspondientes a las palabras “come” (figura 3.6) y “comida” (figura 3.7) serán similares, las dos se basan en la entrada del verbo “comer”. En el caso (1) el sujeto es “el niño” y el objeto es “la manzana” mientras que en el caso (2) ocurre lo inverso, sin embargo la estructura argumental de las mismas es análoga. Esto se representa en las entradas léxicas como se muestra en las figuras. En el caso de “come”, el especificador está coindizado con el argumento 0 (agente), y el complemento con el argumento 1 (tema). En el caso

de “comida” el complemento está coindizado con el agente y el especificador con el tema.

$$\begin{bmatrix}\text{word} & \\ \text{SYN} & \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{v} & \\ \text{AGR} & \begin{bmatrix}\text{FORM} & \text{personal} \\ \text{MODE} & \text{i} \\ \text{PER} & 3 \\ \text{NUM} & \text{sg}\end{bmatrix}\end{bmatrix} \\ \text{VAL} & \begin{bmatrix}\text{SPEC} & \langle \boxed{1}\ \text{SYN.HEAD n}\rangle \\ \text{COMPS} & \langle \boxed{2}\ \text{SYN.HEAD n}\rangle\end{bmatrix}\end{bmatrix} \\ \text{SEM} & \begin{bmatrix}\text{ARGS} & \begin{bmatrix}\text{ARG0} & \boxed{1} \\ \text{ARG1} & \boxed{2}\end{bmatrix}\end{bmatrix} \\ \text{TEXT} & \textit{come}\end{bmatrix}$$

Figura 3.6: Estructura de rasgos de “come”

$$\begin{bmatrix}\text{word} & \\ \text{SYN} & \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{v} & \\ \text{AGR} & \begin{bmatrix}\text{FORM} & \text{participio} \\ \text{NUM} & \text{sg} \\ \text{GEN} & \text{f}\end{bmatrix}\end{bmatrix} \\ \text{VAL} & \begin{bmatrix}\text{SPEC} & \langle \boxed{1}\ \text{SYN.HEAD n}\rangle \\ \text{COMPS} & \langle \boxed{2}\ \text{SYN.HEAD s}\rangle\end{bmatrix}\end{bmatrix} \\ \text{SEM} & \begin{bmatrix}\text{ARGS} & \begin{bmatrix}\text{ARG0} & \boxed{2} \\ \text{ARG1} & \boxed{1}\end{bmatrix}\end{bmatrix} \\ \text{TEXT} & \textit{comida}\end{bmatrix}$$

Figura 3.7: Estructura de rasgos de “comida”

3.2.2 Frase

Las frases son las estructuras que representan todos los constituyentes o combinaciones de palabras (grupos y sintagmas nominales, grupos verbales, oraciones, etc.). La estructura de rasgos para una frase (tipo `phrase`) se muestra en la figura 3.8.

$$\left[\begin{array}{lll} \text{phrase} & & \\ \text{SYN} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll} \text{n} \mid \text{v} \mid \ldots & \\ \text{AGR} & \left[\begin{array}{l} \text{PER } 1 \mid 2 \mid 3 \\ \text{NUM sg} \mid \text{pl} \\ \text{GEN f} \mid \text{m} \\ \ldots \end{array}\right] \end{array}\right] \\ \text{VAL} & \left[\begin{array}{ll} \text{SPEC} & \langle \text{expr} \rangle \\ \text{COMPS} & \langle \text{expr} \rangle \end{array}\right] \\ \text{SYNHEAD} & \text{expr} \end{array}\right] \\ \text{SEM} & \left[\text{SEMHEAD} \quad \text{expr}\right] \end{array}\right]$$

Figura 3.8: Estructura de rasgos para una frase

Los rasgos que se utilizan son muy similares a los de las palabras (heredados del tipo `expr`), con un par de diferencias importantes. Todas las frases son estructuras que contienen uno o más hijos en el árbol sintáctico. De estos hijos, la teoría indica que uno debe estar marcado como núcleo sintáctico. Esto se indica mediante el rasgo `SYNHEAD` que está dentro del rasgo `SYN`. Este rasgo está relacionado con el rasgo `HEAD`, pero no es exactamente igual ya que `HEAD` busca modelar los atributos morfosintácticos que permiten controlar la concordancia con el resto de la frase, mientras que `SYNHEAD` es simplemente un puntero a la expresión que actúa como núcleo sintáctico de la frase. En algunos casos podría ocurrir que los rasgos `SYN.HEAD` y `SYN.SYNHEAD` no concordaran, por ejemplo: al construir una coordinación de elementos singulares (`NUM=sg`) es deseable que el número de la coordinación concuerde con plurales (`NUM=pl`).

Además del núcleo sintáctico, se decidió modelar un nuevo rasgo denominado núcleo semántico (rasgo `SEMHEAD`). Este rasgo apunta a una expresión que actúa como núcleo de la estructura semántica de la frase. Existe una diferencia importante entre el núcleo sintáctico y el núcleo semántico: el núcleo sintáctico guía la concordancia de la frase con su contexto, mientras que el semántico especifica los argumentos permitidos. Para una gran cantidad de casos los dos núcleos apuntarán a la misma expresión, pero existen situaciones donde los núcleos serán distintos, lo cual se explicará más adelante.

3.3 Reglas de la gramática

La gramática consta solamente de 12 reglas, todas ellas binarias. Cada regla indica cuál de las expresiones que se combinan será el núcleo sintáctico de la frase resultante. En la mayoría de los casos el núcleo semántico será igual al sintáctico, salvo algunas excepciones que se mencionarán en la regla correspondiente.

La mayoría de estas reglas ya están dadas por la teoría: reglas del especificador, complemento y modificador [41]. Además de estas reglas, se agregaron algunas para tomar en cuenta particularidades del corpus con el que trabajamos.

3.3.1 Reglas de especificador

- `spec_head` → `spec head`
- `head_spec` → `head spec`

Dada una expresión que contiene un especificador no saturado, es posible anexar un especificador por la derecha o por la izquierda. La frase resultante tiene el especificador saturado. El rasgo `SYN.HEAD.AGR` de las dos expresiones debe unificar.

Estas reglas se utilizan tanto para modelar los especificadores de sintagmas nominales (figura 3.9) como los sujetos (figura 3.10) dentro de una oración (se considera que el especificador de un sintagma verbal es un sintagma nominal). De esta manera se evita tener reglas particulares para tratar los sujetos de las oraciones. La regla del especificador por la derecha no tiene sentido en el caso de los determinantes, pero sí en el caso de los sujetos.

3.3.2 Reglas de complemento

- `comp_head` → `comp head`
- `head_comp` → `head comp`

Dada una expresión que contiene un complemento no saturado, es posible anexar un complemento por la derecha o por la izquierda. La frase resultante tiene saturado el complemento correspondiente.

En español es posible que existan complementos (por ejemplo complementos verbales) que estén tanto a la derecha como a la izquierda de su núcleo. En HPSG es posible tratar este fenómeno asumiendo que la posición natural en la que aparece un complemento es a la derecha y marcando que existe un *gap* en ese lugar del complemento que es llenado en otro lugar. En este caso optamos por una alternativa más simple: dos reglas simétricas que permiten tratar los dos casos de la misma manera. Por ejemplo, considerar la frase "Juan come arroz" y su equivalente semántico "arroz come Juan". Esta segunda frase no presenta la estructura canónica del español, que se caracteriza por ser de tipo

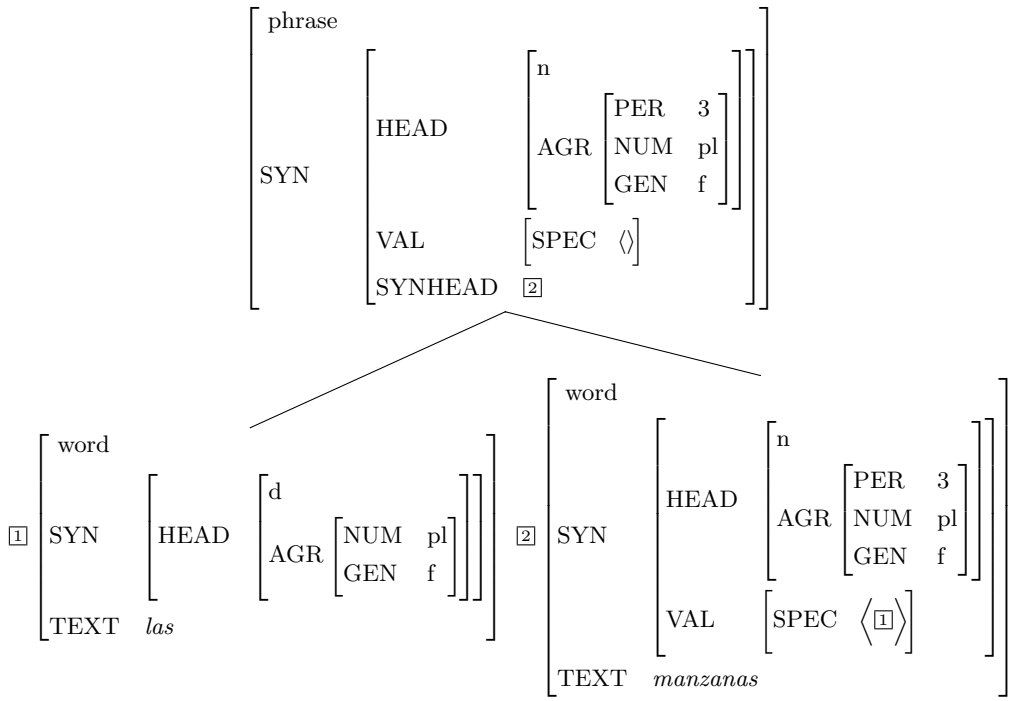

Figura 3.9: Aplicación de la regla `spec_head` para la frase “las manzanas”

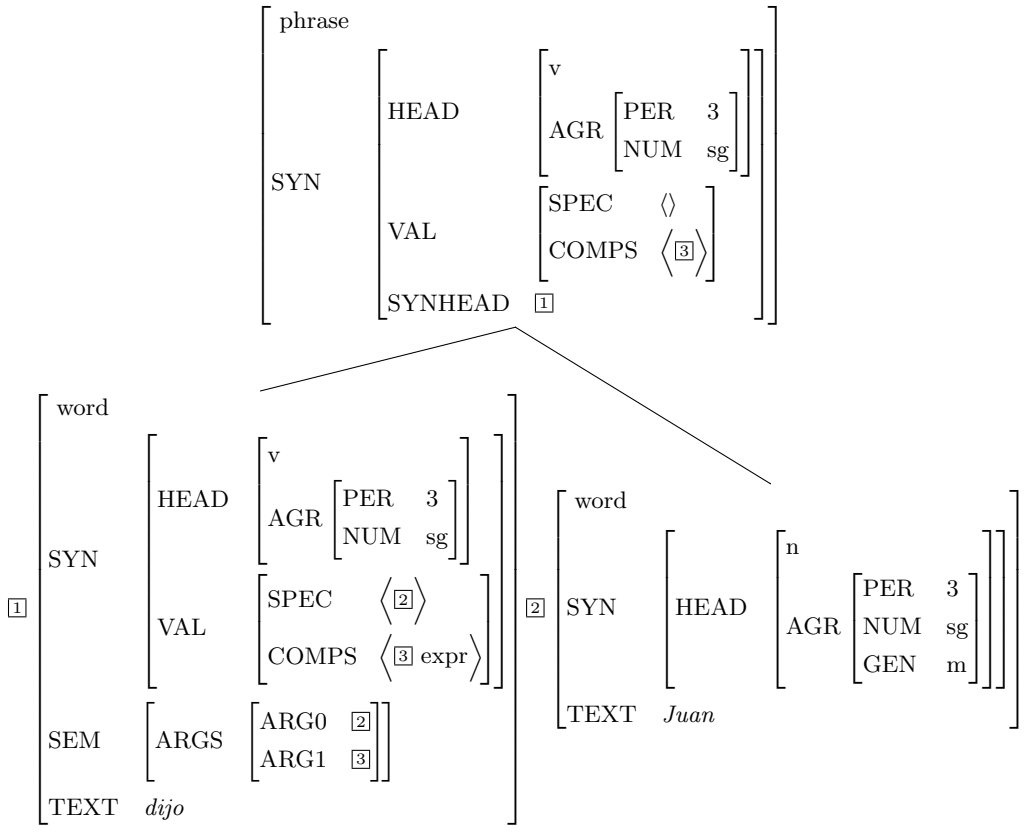

Figura 3.10: Aplicación de la regla `head_spec` para la frase “dijo Juan”

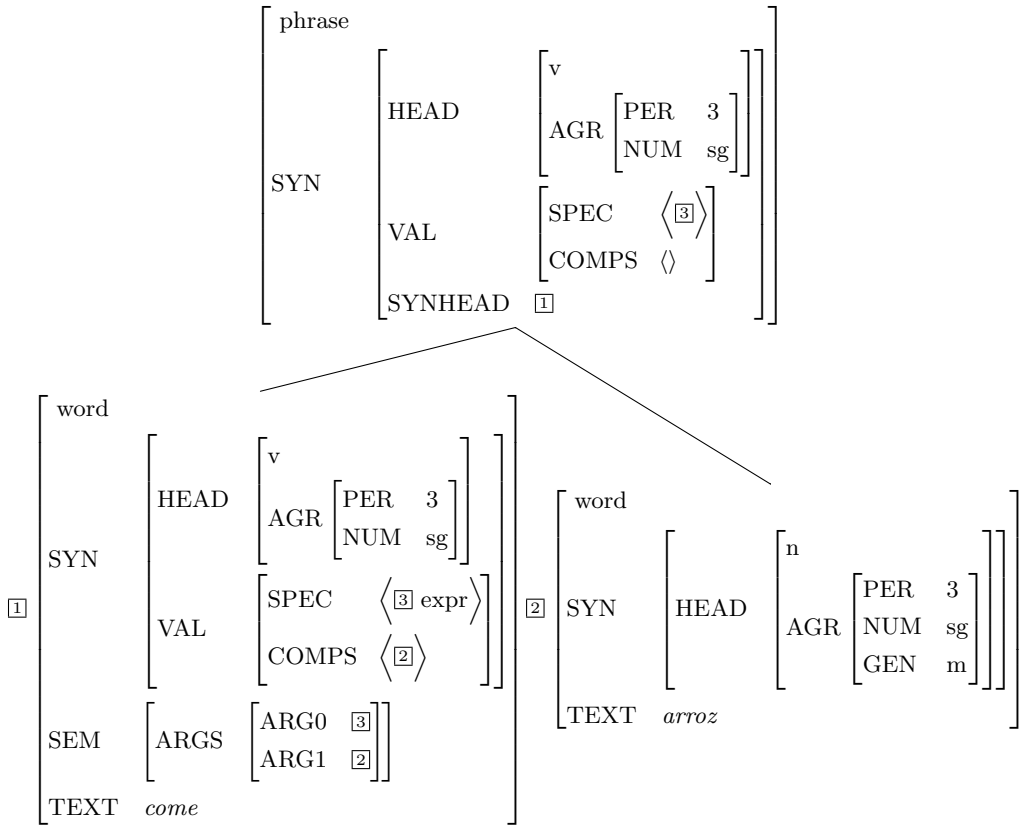

Figura 3.11: Aplicación de la regla `head_comp` para la frase "come arroz"

SVO[1], sin embargo un hablante nativo de la lengua no tendría problema en entenderla e identificar su significado. La figura 3.11 muestra el subárbol que se arma para la frase "come arroz" dentro de "Juan come arroz", mientras que la figura 3.12 muestra el subárbol que se arma para la frase "arroz come" dentro de "arroz come Juan".

En algunos casos de aplicación de la regla del complemento es posible que el núcleo semántico difiera del sintáctico. Considerar por ejemplo la frase "Juan puede comer manzanas". En este caso la oración contiene la perífrasis verbal "puede comer". La frase resultante en el árbol debe poder combinarse con "manzanas" que es el objeto directo de "comer", pero no de "puede". Además, "Juan" es el sujeto de la oración y es sujeto de "puede" (concuerda con el verbo), pero también actúa como el agente para el verbo "comer". Consideramos en-

[1]En los lenguajes SVO las oraciones canónicas se componen de Sujeto-Verbo-Objeto, en ese orden. El inglés y el español son lenguajes SVO, pero en el caso del español es posible realizar algunas variantes de las oraciones que aún mantengan el significado.

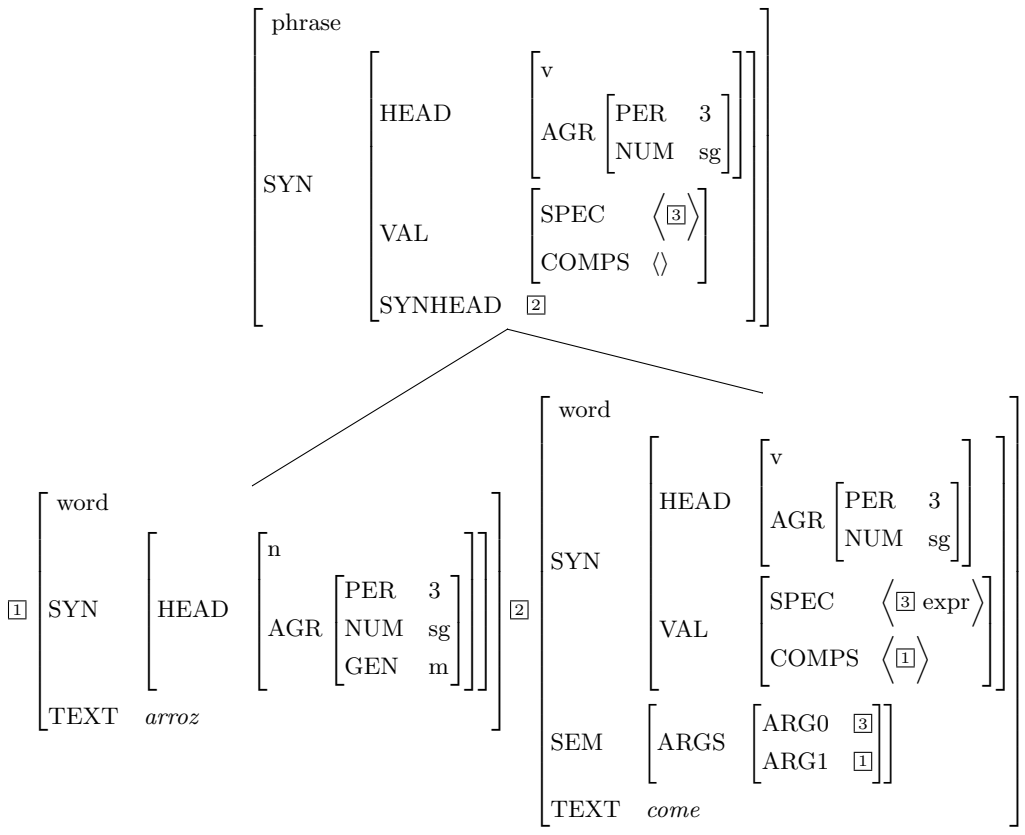

Figura 3.12: Aplicación de la regla `comp_head` para la frase “arroz come”

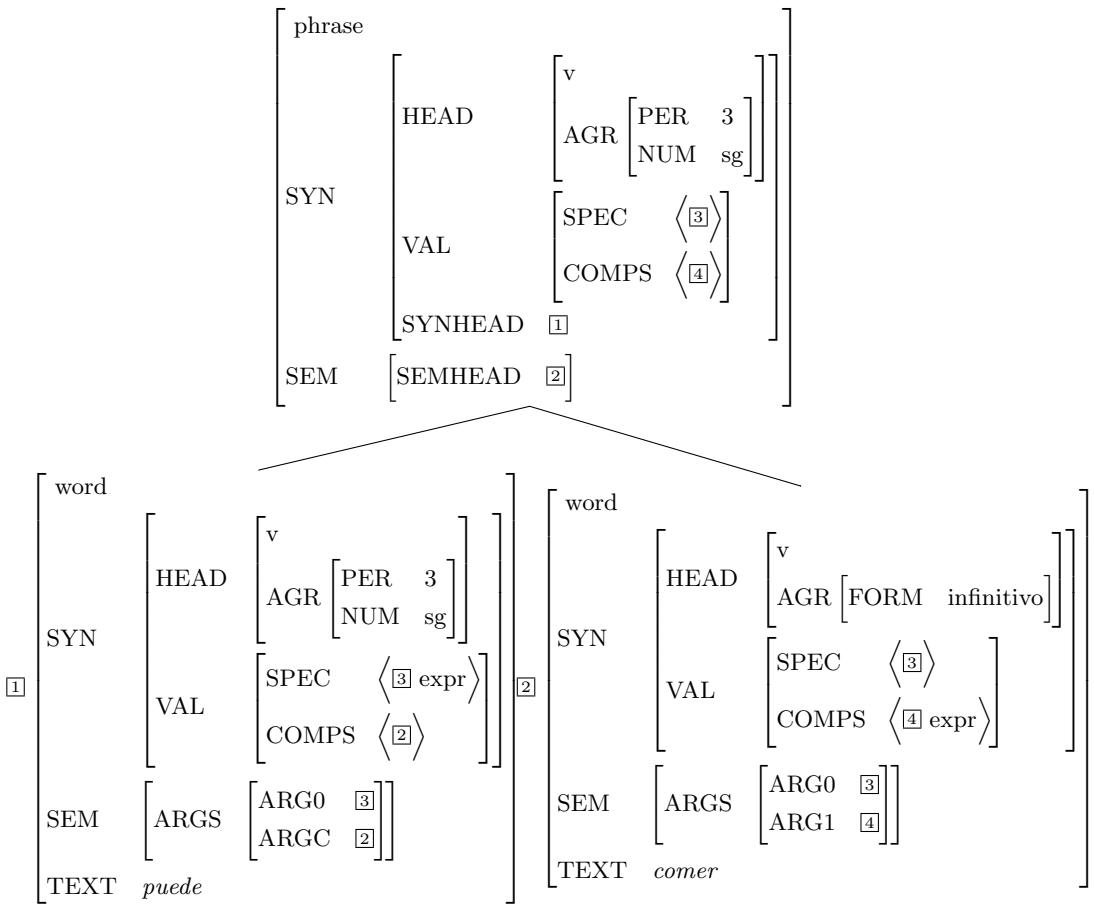

Figura 3.13: Aplicación de la regla `head_comp` para la frase "puede comer"

tonces que en esta frase hay un núcleo sintáctico que establece restricciones de concordancia, y un núcleo semántico que establece cuáles son los argumentos a seleccionar. En la sección 4.4.7 se explica más en detalle la transformación de este tipo de estructuras verbales. La figura 3.13 muestra el ejemplo de construcción de la frase "puede comer" mediante la regla `head_comp`.

3.3.3 Reglas de modificador

- `mod_head` → `mod head`
- `head_mod` → `head mod`

Dada una expresión que tiene un indicador de modificador no saturado, es posible anexarla a otra expresión modificada por la izquierda o por la derecha.

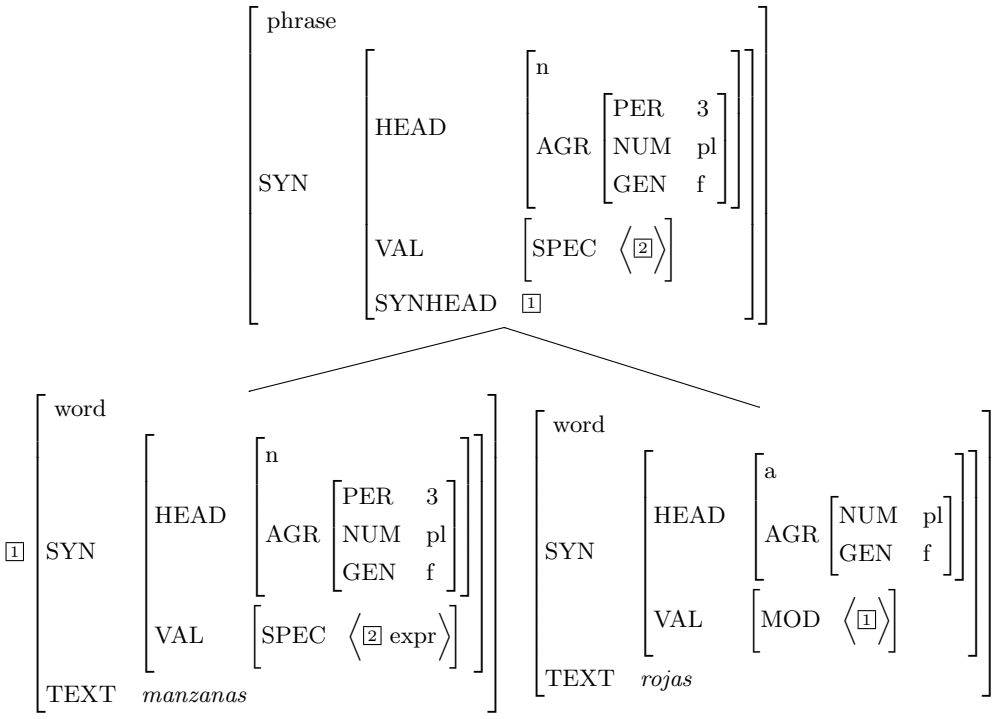

Figura 3.14: Aplicación de la regla `head_mod` para la frase "manzanas rojas"

La frase resultante tiene el rasgo de modificador (`MOD`) vacío. Tanto el núcleo sintáctico como el semántico de la frase resultante apuntan a la expresión modificada.

Las figuras 3.14 y 3.15 son ejemplos de aplicaciones de las reglas de modificador por la derecha y por la izquierda respectivamente.

3.3.4 Coordinaciones

Las coordinaciones en general no siguen reglas binarias, las reglas son por lo menos ternarias debido a que se necesitan los dos elementos coordinados más una conjunción o un símbolo de puntuación en medio. Puede haber incluso más elementos en la regla, no está limitada la cantidad de constituyentes que pueden estar coordinados en una sola estructura. Sin embargo, el objetivo es pasar todas las estructuras a reglas que sean a lo sumo binarias. Se utilizan dos reglas para modelar una coordinación ternaria: `coord_left` y `coord_right`.

- `coord_right` → `conj expr`
- `coord_left` → `expr coord_right`

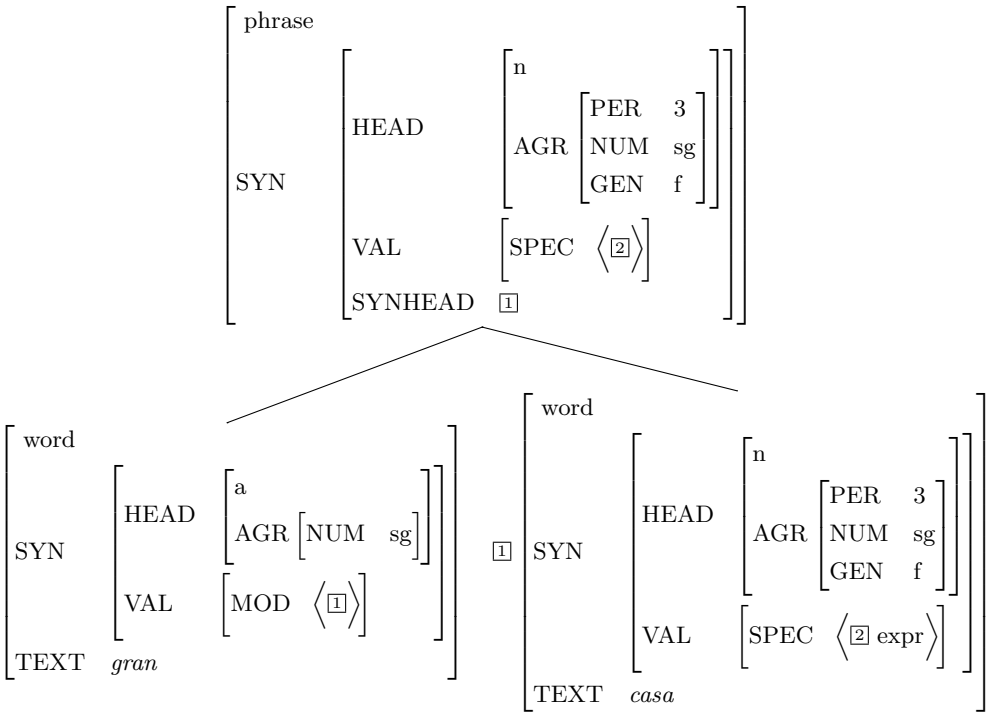

Figura 3.15: Aplicación de la regla `mod_head` para la frase "gran casa"

La coordinación más simple posible incluye un elemento a la izquierda, una conjunción y un elemento a la derecha, por ejemplo la de la figura 3.16.

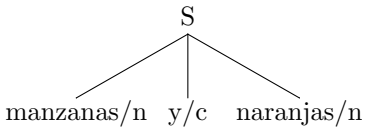

Figura 3.16: Coordinación simple con dos elementos

Para escribir esto utilizando las dos reglas binarias, se considera que primero se aplica la regla `coord_right` (que se aplica a una conjunción con un elemento a la derecha) y luego la regla `coord_left` (que se aplica al elemento de la izquierda y el resultado de la regla `coord_right`). El árbol resultante queda como se muestra en la figura 3.17.

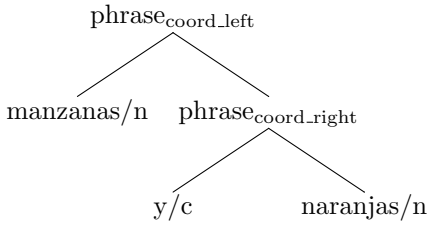

Figura 3.17: Coordinación con dos elementos binarizada

Para coordinaciones de tamaño mayor, es posible iterar la aplicación de estas reglas para cubrir todos los elementos necesarios. Se considera que la secuencia de elementos coordinados siempre deberá estar separada por conjunciones o elementos análogos como el símbolo de puntuación coma (","). Considerar el ejemplo de la figura 3.18. La versión binarizada de este ejemplo se muestra en la figura 3.19.

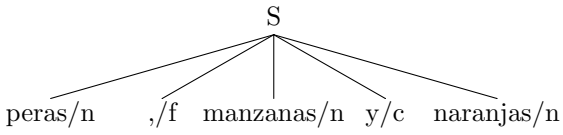

Figura 3.18: Coordinación con tres elementos

3.3.5 Reglas simplificadas

Las siguientes reglas se agregaron para reducir la complejidad del análisis de ciertas construcciones lingüísticas que se decidió dejar fuera del alcance de este trabajo. El objetivo de estas reglas es permitir detectar y dejar marcado en el corpus la presencia de estas construcciones, para que a futuro se puedan

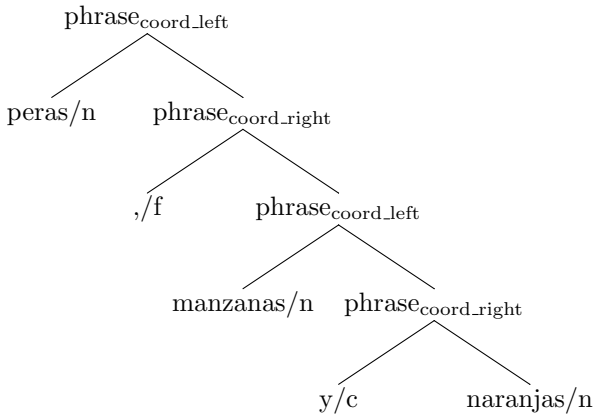

Figura 3.19: Coordinación con tres elementos binarizada

identificar fácilmente y proceder a realizar un análisis más profundo y más rico donde se incorpore toda la información necesaria. Son las reglas para tratar la puntuación, los clíticos y las relativas.

3.3.5.1 Reglas de puntuación

- `punct_head` → `punct head`
- `head_punct` → `head punct`

Dada una expresión cualquiera y un símbolo de puntuación, es posible anexar el símbolo de puntuación a la derecha o a la izquierda de la expresión. La aplicación de símbolos de puntuación siempre está permitida y no hay restricciones respecto a esto. El núcleo siempre es la expresión y la combinatoria de la frase se mantiene igual que la del núcleo.

Esta regla está pensada para absorber los símbolos de puntuación que no están participando de las coordinaciones. Pero además de las coordinaciones, los símbolos de puntuación en español también participan de otro tipo de construcciones. En el presente trabajo no se realiza un análisis de otro tipo de estructuras con símbolos de puntuación y por simplicidad se elige absorberlos junto con el constituyente que acompañan.

3.3.5.2 Regla del clítico

- `clitic_head` → `clitic head`

Los clíticos en español son pronombres que pueden actuar como representantes de los argumentos verbales. En algunos casos la presencia del clítico sustituye a la del argumento verbal, pero en otros casos en español es común que aparezcan tanto el argumento real como el clítico, lo que se conoce como duplicación de clíticos [43]. Los clíticos pueden aparecer antes del verbo como

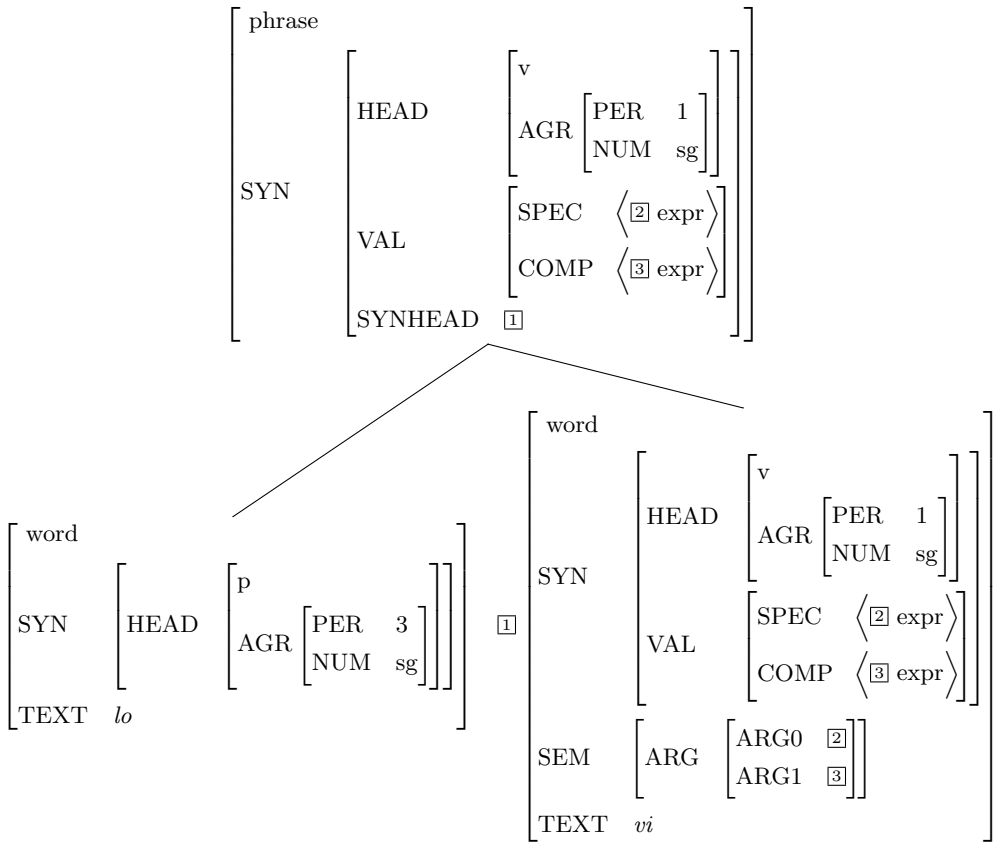

Figura 3.20: Aplicación de la regla `clitic_head` para la frase "lo vi" dentro de "lo vi a Juan"

palabras independientes ("yo **lo** vi **a Juan**") o pueden anexarse en una posición sufija del verbo ("iba a ver**lo**"). La regla `clitic_head` se encarga de modelar el caso en que el clítico es una palabra independiente que aparece a la izquierda del verbo.

Los clíticos actúan como complementos, pero para evitar la discrepancia entre los clíticos que sustituyen al argumento verbal y los que lo duplican, se toma la decisión de crear una regla separada de la de los complementos para tratarlos. Esta regla no satura uno de los complementos que está esperando el verbo, por lo que si se necesita hacer un análisis de la estructura argumental que incluya los clíticos será necesario un procesamiento extra del árbol para descubrir cuáles clíticos están ocupando lugares de argumentos y a qué otras expresiones apuntan.

La figura 3.20 muestra un ejemplo de aplicación de la regla del clítico para la frase "lo vi" dentro de "yo lo vi a Juan".

Un análisis más correcto de estas estructuras debería tener en cuenta los dos posibles usos: clítico duplicado o pronombre que actúa como argumento. En el primer caso el clítico puede ser absorbido, idealmente dejando una referencia que apunta al verdadero argumento que también estará presente en la oración. En el segundo caso el clítico deberá tomar el lugar del argumento en la estructura de rasgos, lo cual implica cierta transformación intermedia para que los rasgos del clítico unifiquen con los rasgos que esperaba el verbo. Se decidió dejar afuera del alcance de este trabajo la complejidad de la resolución de estos casos, por lo que la regla simplemente los absorbe y los deja marcados para en un futuro poder clasificar y analizar apropiadamente los dos tipos de clíticos.

3.3.5.3 Regla de relativa

- `head_rel` → `head rel`

Los pronombres relativos son pronombres que permiten la introducción de una oración (subordinada) dentro de otra oración. Por ejemplo los pronombres "que" y "donde" se utilizan en las frases "la película que vi" o "la casa donde vivo". Además de pronombres, existen ejemplos con expresiones de varias palabras que pueden introducir una oración relativa (por ejemplo "en donde" y "cuyo" seguido de una frase nominal). A todos los pronombres y expresiones que pueden introducir una oración relativa los denominaremos *expresiones pronominales relativas*. La oración relativa que introducen puede tener una dependencia con el contexto, y en esos casos la expresión pronominal relativa actúa como uno de los argumentos dentro de la oración, además de ser el nexo con la oración contenedora. Estas oraciones relativas pueden actuar como modificadores de nombres, por ejemplo: "el libro **que compré**".

El tratamiento de las expresiones relativas es un subconjunto de un problema más grande que es el de las dependencias de largo alcance. En la teoría HPSG se suelen utilizar construcciones de tipo `filler-gap` para tratar estas dependencias. En el presente trabajo se decidió dejar fuera del alcance el problema de las dependencias de largo alcance introducidas por expresiones relativas, mientras que otros ejemplos de dependencias de largo alcance sí son modelados con un enfoque distinto, por ejemplo: la coindización de los argumentos verbales en los casos de perífrasis.

La regla `head_rel` permite combinar una expresión pronominal relativa con la oración relativa que introduce. Esta regla existe para marcar la introducción de oraciones relativas, pero no se realiza ninguna identificación de rasgos más profundos, por ejemplo no se indica como qué argumento actúa la expresión pronominal relativa dentro de la oración relativa, ni a qué entidad fuera de la oración relativa apunta el pronombre.

La figura 3.21 muestra un ejemplo de aplicación de la regla de relativa para la frase "que compré" dentro de "el libro que compré". En este caso la frase resultante se utilizará como modificador, por lo que se utiliza la versión del pronombre "que" que puede actuar como modificador de un sintagma nominal. Notar que los argumentos no saturados de "compré" no pasan a la frase, por

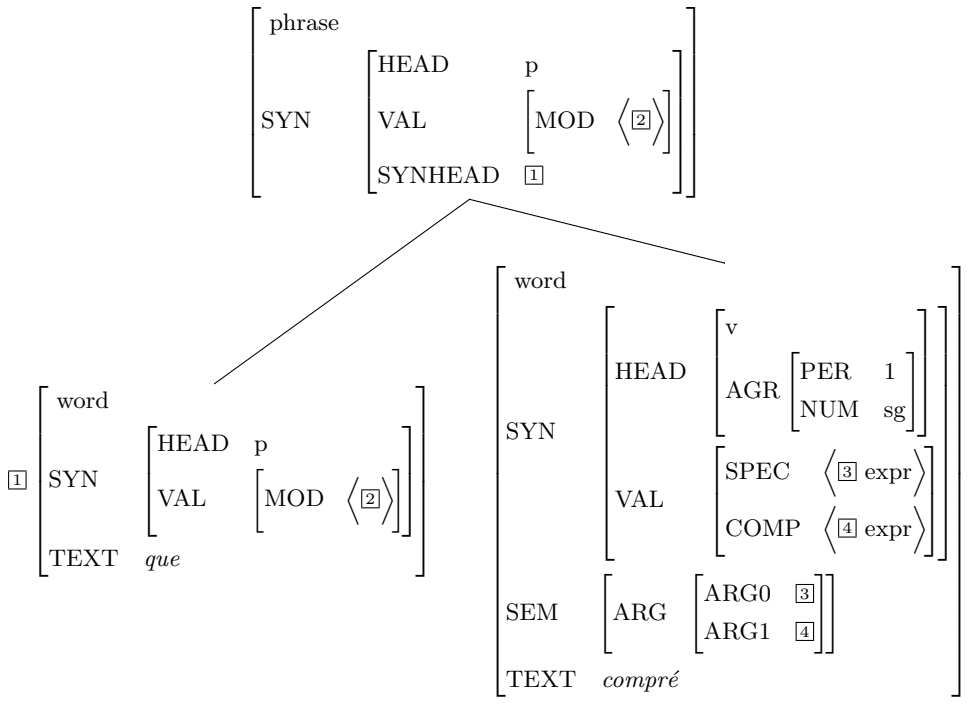

Figura 3.21: Aplicación de la regla `head_rel` para la frase "que compré"

lo que el verbo de la oración relativa queda aislado del contexto de la oración contenedora y no se modelan las dependencias de largo alcance.

Se decidió dejar fuera del alcance del presente trabajo el tratamiento profundo de las relativas así como la identificación de dependencias de largo alcance. La regla `head_rel` se limita a unir una expresión pronominal relativa con la oración relativa que introduce. Teniendo estas construcciones marcadas en el corpus habilita a que en un futuro se pueda realizar un procesamiento más apropiado de las mismas.

Capítulo 4

Transformación del corpus

Este capítulo describe el proceso de transformación del corpus AnCora de su formato original (una gramática libre de contexto enriquecida con atributos) a un formato que cumpla con lo establecido por nuestra gramática HSPG. Se muestra también una evaluación del proceso de transformación del corpus.

4.1 Proceso de transformación

Podemos ver el corpus AnCora como un corpus de árboles sintácticos anotados mediante una gramática libre de contexto. En una gramática HPSG, por el contrario, se intenta que las reglas sintácticas sean lo más simples posibles (regla de especificador, regla de complementos, etc.). Por lo tanto se optó por una estrategia que reduzca la complejidad de las reglas del corpus, llevándolas todas a reglas binarias o unarias. La estrategia adoptada es una transformación que combina un enfoque *top-down* con un enfoque *bottom-up*.

Consideramos que un tipo de árbol HSPG simple o elemental es un árbol que está compuesto solamente por un núcleo rodeado de sus complementos, modificadores y adjuntos. El proceso *top-down* detecta todos los constituyentes de AnCora que no caen dentro de esta categoría y los transforma a una composición de árboles simples. Para cada uno de estos árboles simples se ejecuta el proceso *bottom-up*, que asume como entrada árboles en este formato e identifica los núcleos y las diferentes reglas que se aplican para formar el árbol.

4.2 Estructura de AnCora

Considerándolo como gramática libre de contexto, la cantidad de reglas de dicha gramática es muy grande. AnCora contiene 34 tipos de constituyentes sintácticos (por ejemplo `grup.nom`, `S`, `grup.verb`, `sp`), y se utilizan múltiples reglas en la anotación de cada uno de ellos. La tabla 4.1 muestra la cantidad de reglas que se utilizan para cada constituyente sintáctico. Por ejemplo: 5.800

Constituyente	Reglas
S	5826
sentence	2403
grup.nom	905
sn	295
sp	160
s.a	128
grup.a	112
spec	102
sa	80
sadv	69
grup.verb	68
grup.adv	37
infinitiu	28
conj	12
inc	12
prep	11
interjeccio	9
gerundi	7
relatiu	5
participi	4
neg	4
morfema.verbal	1
morfema.pronominal	1

Tabla 4.1: Cantidad de reglas para cada constituyente en AnCora

formas de escribir una oración subordinada y 900 formas de escribir un grupo nominal.

Los elementos de AnCora se pueden separar en dos grandes grupos: terminales y no terminales. Los elementos de tipo terminal representan las palabras y nunca tienen elementos hijos. Sí contienen por ejemplo la siguiente información:

- Tipo de nodo: por ejemplo `n` (nombre), `s` (preposición), `v` (verbo), etc.
- Atributos: información extra sobre la palabra, en particular nos interesan:
  - `id`: Identificador único del elemento (este atributo no está presente originalmente en AnCora, se incluyó mediante un preprocesamiento del corpus para poder trabajar de manera más cómoda).
  - `wd`: String de la palabra completa.
  - `lem`: Lema de la palabra.
  - `pos`: Información morfosintáctica completa de la palabra, utilizando la notación de etiquetas Eagles [44].
  - `deverbal`: Indica si la palabra es un nombre deverbal, como por ejemplo: “declaración”.

Los elementos de tipo no terminal incluyen todos los constituyentes anotados en el corpus y pueden contener como hijos una combinación de terminales y no terminales. Son elementos XML que contienen la siguiente información:

- Tipo de nodo: por ejemplo `grup.nom`, `sn`, `spec`, etc.
- Atributos: información extra sobre el constituyente. Existen muchos tipos de atributos y son diferentes para cada tipo de constituyente. Tampoco están definidos todos los atributos relevantes para todos los constituyentes. Se puede encontrar por ejemplo:
  - `id`: Identificador único del elemento (análogo al caso anterior, este atributo se agregó en el preprocesamiento del corpus).
  - `func`: Función sintáctica del constituyente (esto es respecto a la estructura que lo contiene), por ejemplo sujeto, complemento directo, etc. No está presente en todos los casos.
  - `arg`: Indica el valor que tiene el constituyente como argumento de la estructura semántica en una notación análoga a PropBank (por ejemplo `arg0`, `arg1`, `argM`, etc.).
  - `coord`: Indica si el constituyente contiene una coordinación. A veces demarca que el constituyente es una coordinación (por ejemplo `grup.nom58`: "Río_Bravo y Saltillo"). Otras veces que hay una coordinación anidada en algún lugar dentro de la estructura del constituyente (por ejemplo `grup.nom57`: "Río_Bravo y Saltillo para la compañía francesa"), no está claro por qué se marcan de esta manera estos constituyentes en el corpus en vez de marcarlo solamente para los que son una coordinación.

4.3 Proceso top-down

El proceso *top-down* intenta partir todos los constituyentes más complejos de manera de llevarlos a una composición de casos más simples. Los constituyentes considerados complejos son los que incluyen coordinaciones, relativas, subordinadas o símbolos de puntuación. Se realizan procesamientos particulares para estos tipos de estructuras, que se detallan a continuación.

4.3.1 Marcado de clíticos

Como se mencionó anteriormente, se realiza un procesamiento especial para los clíticos. La primera etapa del proceso *top-down* trata de marcar los clíticos y otros morfemas ("se") para que puedan ser identificador por las reglas *bottom-up* en una etapa posterior.

Todas las instancias de pronombres que pertenecen a una lista (se, me, te, le, la, lo, nos, os, les, las, los) se marcan con un atributo `pronoun=morfema` o `pronoun=clitico` dependiendo de si están en un constituyente de tipo morfema o de tipo nominal.

4.3.2 Extracción de bloques

Se extraen como bloques (una nueva categoría denominada `block`) las estructuras que aparecen entre paréntesis, entre signos de interrogación y entre signos de exclamación. Considerar la oración subordinada `S4170` que se muestra en la figura 4.1. Este constituyente contiene un bloque interrogativo, por lo que luego de este paso quedará como se muestra en la figura 4.2.

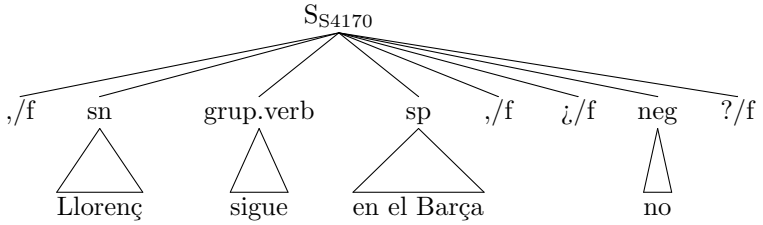

Figura 4.1: Oración subordinada `S4170` original

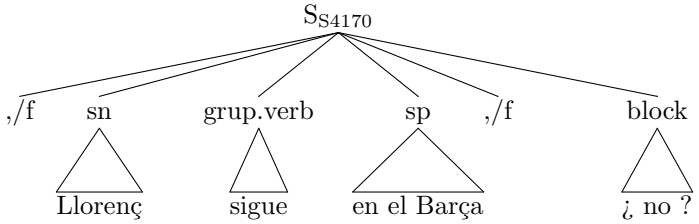

Figura 4.2: Oración subordinada `S4170` luego de la extracción de bloques

El objetivo de este paso es identificar estos nodos complejos y dejarlos separados, ya que el siguiente paso se encargará de extraer todas las puntuaciones al inicio y al final de un bloque para simplificarlo.

Un caso especial está dado por las elipsis de omisión "(. . .)", que se utiliza para omitir texto dentro de una cita. En AnCora estas estructuras aparecen como anotaciones planas, lo cual dificulta el análisis. Considerar el siguiente ejemplo de la oración subordinada `S962`. Este constituyente contiene catorce hijos, donde la mayoría son símbolos de puntuación, como se puede apreciar en la figura 4.3.

Dentro de estos constituyentes, aparece dos veces la estructura "(. . .)", pero se construye esta estructura utilizando los símbolos de puntuación por separado. Este fenómeno aparece varias veces en el corpus. Se extraen estos bloques y se los marca como un constituyente con la categoría `block`, como se muestra en la figura 4.4.

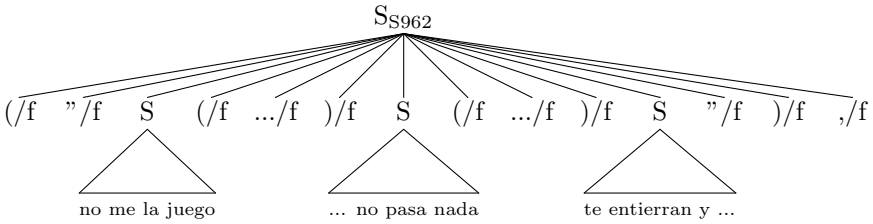

Figura 4.3: Oración subordinada `S962` original

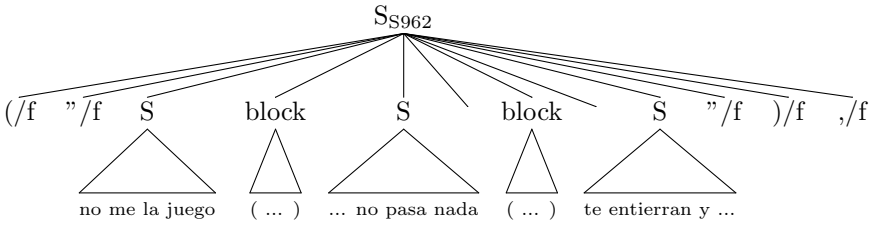

Figura 4.4: Oración subordinada `S962` luego de la extracción de bloques

4.3.3 Extracción de puntuación

Se transforman los constituyentes que tienen símbolos de puntuación al inicio o al final, de manera que el contenido del constituyente (sin los símbolos) quede en un único nodo. Por ejemplo, considerar el sintagma nominal `sn27220`. Originalmente en AnCora está anotado como se muestra en la figura 4.5. Luego del proceso *top-down* quedaría como se muestra en la figura 4.6.

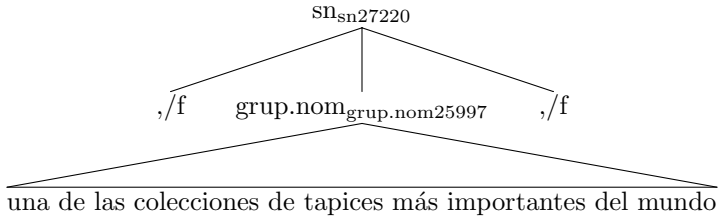

Figura 4.5: Sintagma nominal `sn27220` original

El proceso ya no modificará las dos frases exteriores, solamente continuará procesando el nodo inferior, que contiene el constituyente importante. Los nodos exteriores utilizan las reglas de aplicación de puntuación: `punct_head` y `head_punct`.

Notar que los bloques identificados por el paso anterior del proceso serán transformados en estructuras más simples en este paso, dejando los signos de puntuación de inicio y de fin (pueden ser comas, símbolos de interrogación, símbolos de exclamación, etc.) separados en su propio constituyente.

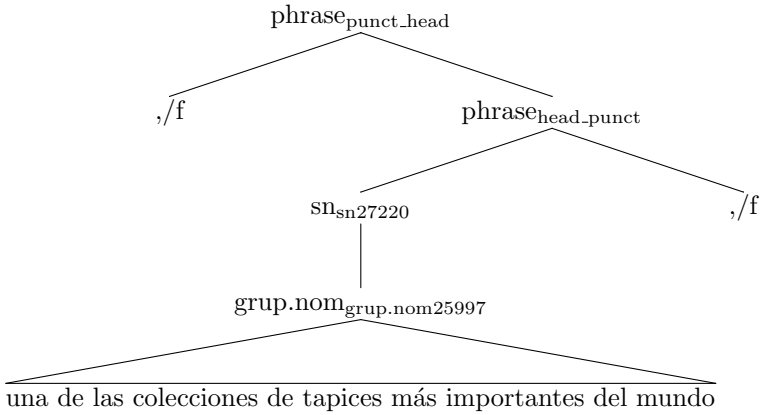

Figura 4.6: Sintagma nominal `sn27220` luego de la extracción de puntuación

4.3.4 Sintagmas nominales que actúan como expresiones temporales

En AnCora existe un conjunto de expresiones temporales que están marcadas consistentemente como sintagmas nominales. Son todas expresiones con la forma "hace <*unidad de tiempo*>", por ejemplo: "hace un año". La estructura en AnCora siempre es plana, como lo muestra el ejemplo de la figura 4.7 del sintagma nominal `sn6747`.

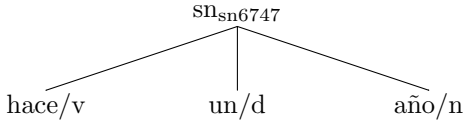

Figura 4.7: Sintagma nominal `sn6747` original

Sintácticamente esto contiene un núcleo verbal, por lo que sería más parecido a una oración subordinada, actuando como un modificador (una estructura de tipo adverbial). Este paso de la transformación convierte estas estructuras en sintagmas adverbiales de tiempo. La figura 4.8 muestra el resultado de la transformación del sintagma nominal `sn6747`.

4.3.5 Incisos

Los constituyentes tipo `inc` en AnCora demarcan incisos. Los incisos pueden aparecer en cualquier lugar dentro de un constituyente, y resulta difícil identificar a dónde deberían estar asociados. Durante la transformación se tomó como heurística que un inciso actúa como un modificador del elemento que lo precede inmediatamente. Por ejemplo, la oración `sentence5072` en AnCora está estructurada como se muestra en la 4.9.

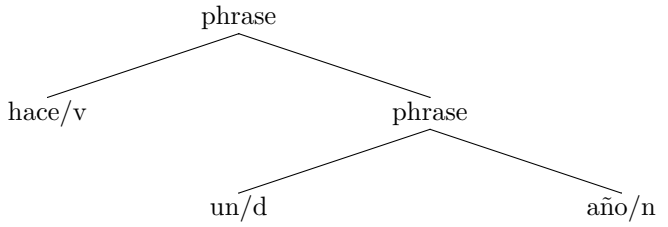

Figura 4.8: Sintagma nominal `sn6747` luego de la transformación de la expresión temporal

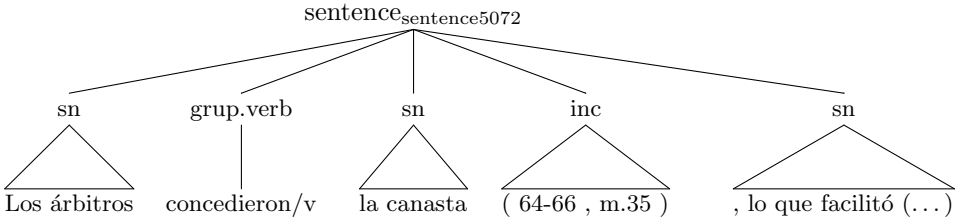

Figura 4.9: Oración `sentence5072` original

No queda claro sobre qué elemento se debería aplicar este inciso. La heurística seleccionada simplemente lo anota como modificador del sintagma nominal anterior. Luego de la transformación la estructura pasa a ser como se indica en la 4.10. La estructura resultante es más simple porque ya no hay que preocuparse por analizar el comportamiento del inciso, ya no será tenido en cuenta en el resto de la transformación. A esto lo denominamos *colapso de incisos*.

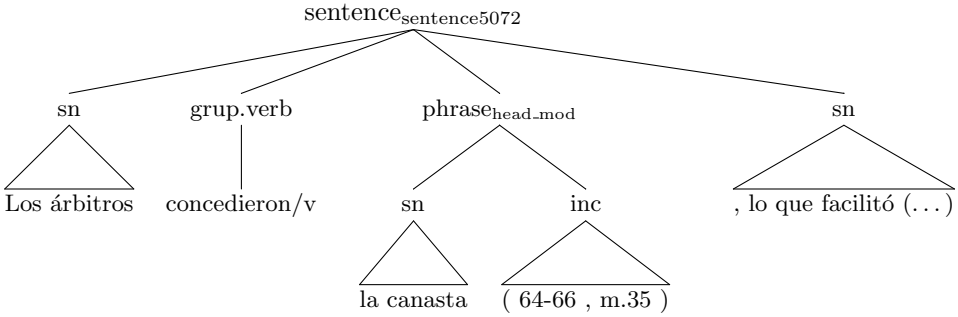

Figura 4.10: Oración `sentence5072` luego del colapso de incisos

4.3.6 Subordinadas y conjunciones al inicio

Considerar la oración subordinada `S3` que se muestra en la figura 4.11.

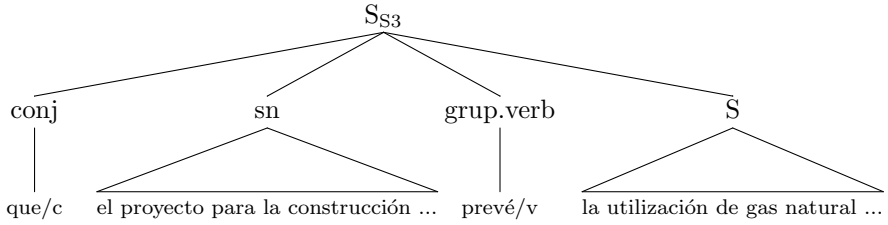

Figura 4.11: Oración subordinada `S3` original

Esta es una oración subordinada de tipo completiva y comienza con una conjunción subordinante "que". Este "que" no cumple ninguna otra función dentro de la oración subordinada, por lo que se lo extrae en un nodo padre y se deja el resto de los elementos juntos en otro nodo, para poder trabajar con ellos más adelante en la transformación. El nodo raíz de esta estructura utiliza la regla `head_comp`. De esta manera queda la conjunción subordinante como núcleo sintáctico de la estructura, y el resto de la frase como complemento, como lo muestra la figura 4.12.

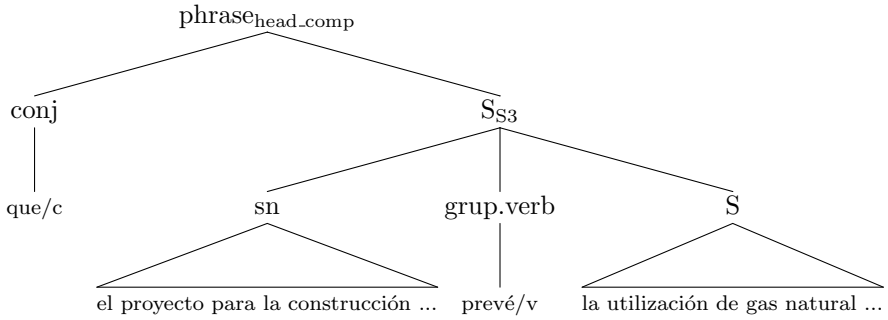

Figura 4.12: Oración subordinada `S3` luego de la extracción de conjunción

Los constituyentes que se inician con una conjunción, pero que no están formando parte de una estructura coordinada, se tratan de la misma manera. Considerar por ejemplo la oración `sentence58` que se muestra en la figura 4.13.

Esta oración comienza con una conjunción "pero", y es la palabra inicial de la oración, por lo que no es la coordinación entre dos frases. Esto puede ocurrir al inicio de una oración o también al inicio de otros constituyentes. Se transforman estos constituyentes para que la conjunción sobrante se encuentre sola y el resto del constituyente forme parte de un solo nodo. El resultado es como el que se muestra en la figura 4.14.

Estos casos se procesan igual para la conjunción subordinante "que", por lo que se aplica la misma regla para la raíz del árbol: la regla `head_comp`. El

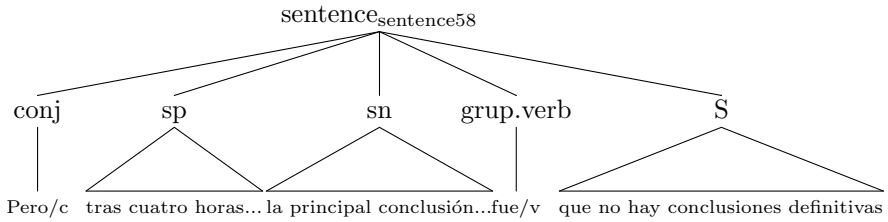

Figura 4.13: Oración `sentence58` original

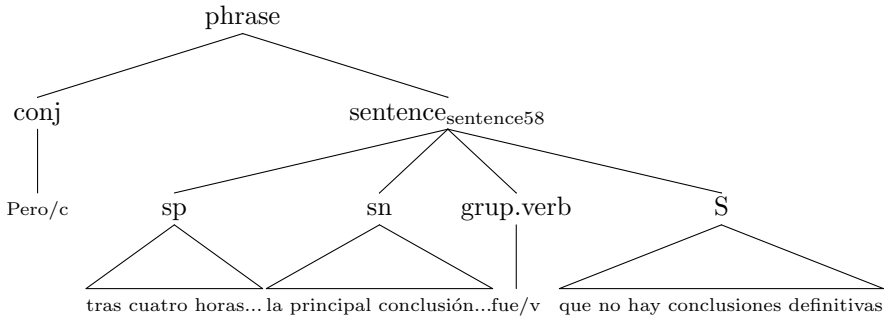

Figura 4.14: Oración `sentence58` luego de la extracción de conjunción

objetivo es que las conjunciones subordinantes queden estructuradas de manera similar a los grupos preposicionales y a los pronombres relativos.

Existe otro caso de oraciones subordinadas que se encuentran en medio de un constituyente. Considerar por ejemplo la oración `sentence615` que se muestra en la figura 4.15.

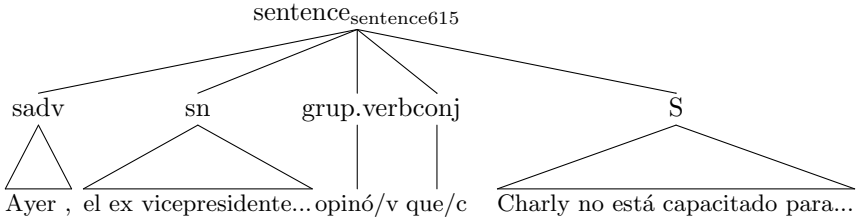

Figura 4.15: Oración `sentence615` original

Esta oración contiene una oración subordinada "que Charly no está capacitado para…". Sin embargo la estructura del constituyente no refleja esta particularidad, la oración subordinada aparece aplanada dentro de un constituyente más grande. El proceso extrae los dos nodos y construye uno nuevo que los incluye. Esta nueva frase también utiliza la regla `head_comp` para combinar a sus hijos, como se muestra en la figura 4.16.

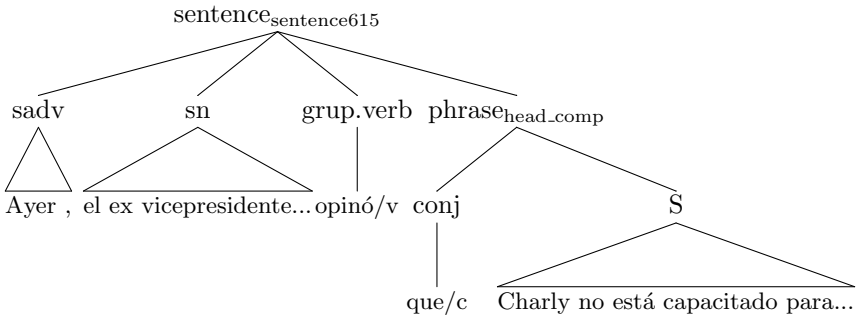

Figura 4.16: Oración `sentence615` luego de la extracción de conjunción

4.3.7 Relativas

El siguiente paso trata de la extracción y marcado de las oraciones relativas. Estas oraciones siempre actúan como modificadores de sintagmas nominales. Por ejemplo la oración `sentence46`: "En otros dos municipios que forman_parte del área exterior de este " cinturón rojo " , Humanes y Valdemoro . . . ". El nombre "municipios" está siendo modificado por la oración relativa "que forman_parte del área exterior de este " cinturón rojo " ". En AnCora esta oración relativa comienza con un elemento de tipo `relatiu`, que contiene el pronombre relativo "que". Sin embargo, como se mencionó en la sección 3.3.5.3, existen otros tipos de estructuras mediante las cuales se puede introducir una relativa. Denominamos a estas estructuras *expresiones pronominales relativas* y se muestran ejemplos en la tabla 4.2.

Tipo de expresión pronominal relativa	Ejemplo
Elemento `relatiu`	S160: [[$_{\text{relatiu}}$ que] [forman_parte] [del área exterior de este " cinturón rojo "]]
Elemento `sadv` que termina en elemento `relatiu`	S22727: [[más [allá [del [$_{\text{relatiu}}$ cual]]]] [está] [el Paraíso]]
Elemento `sp` que termina en elemento `relatiu`	S22875: [[a [la [$_{\text{relatiu}}$ que]]] [Arana] [hace significar] [vasco]]
Elemento `sn` que termina en elemento `relatiu`	S25831: [[la [$_{\text{relatiu}}$ cual]] [aseguró] [que casi ninguno de sus compañeros ha podido dormir]]
Elemento `sn` que comienza con lema "cuyo"	S26013: [[[$_{\text{relatiu}}$ cuyo] objetivo] [sería] [asegurar una paz duradera en la región]]

Tabla 4.2: Ejemplos de relativas

El proceso *top-down* intenta identificar todos los constituyentes que comienzan con una expresión pronominal relativa, extrae dicha expresión y deja una marca que luego el proceso *bottom-up* utilizará para establecer las reglas correc-

tas.

Por ejemplo, la figura 4.17 muestra la estructura de la oración subordinada relativa `S160` en AnCora.

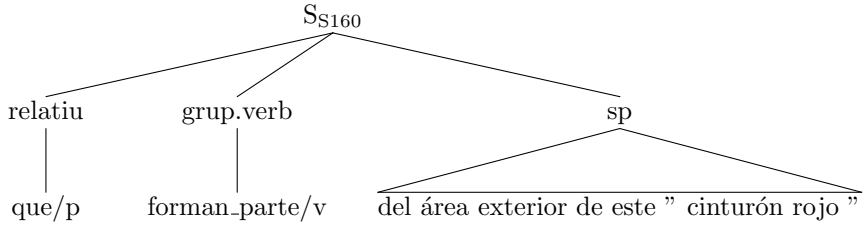

Figura 4.17: Oración subordinada `S160` original

Esta estructura se transformará en la que se muestra en la figura 4.18, dejando una marca en el nodo padre que indica que es una oración relativa.

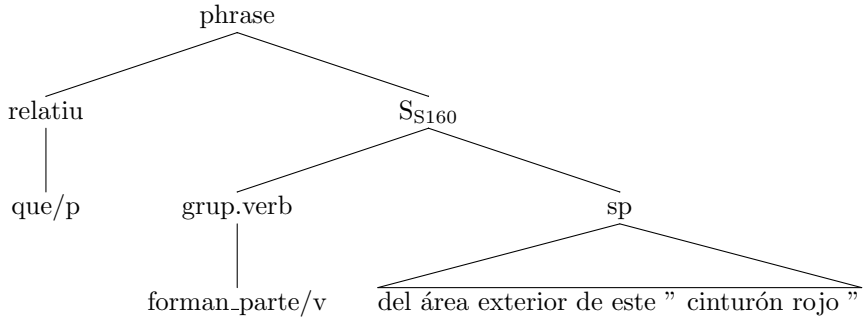

Figura 4.18: Oración subordinada `S160` luego de la extracción de relativas

4.3.8 Preposiciones

Existen algunos casos en AnCora en que un constituyente está marcado como oración subordinada pero comienza con una preposición. Estos casos parecen más errores de etiquetado. En la figura 4.19 se muestra la oración subordinada `S1798` en AnCora. En este paso de la transformación se extrae la preposición al inicio y se lo convierte para que se comporte como un sintagma preposicional normal, como lo muestra la figura 4.20.

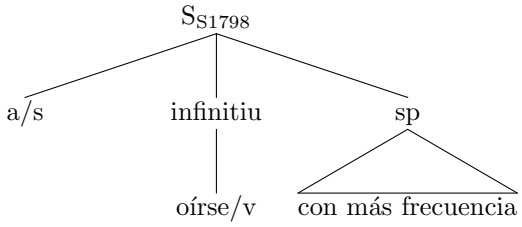

Figura 4.19: Oración subordinada `S1798` original

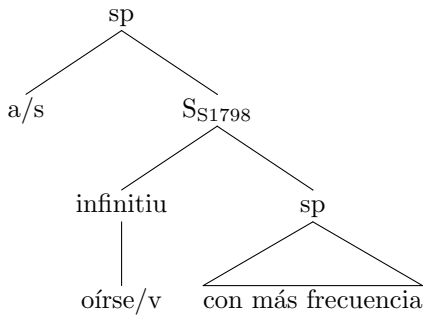

Figura 4.20: Oración subordinada `S1798` luego de la extracción de preposiciones

4.3.9 Coordinaciones

La última etapa del proceso *top-down*, y la de mayor complejidad, es la transformación de las coordinaciones. Como se vio en la sección 3.3.4, las coordinaciones ternarias (o de aridad mayor) se deben descomponer en una sucesión de aplicaciones de las reglas binarias `coord_left` y `coord_right`.

El proceso de transformación de las coordinaciones consta de tres etapas, denominadas:

- Colapso de conjunciones múltiples
- Binarización de secuencias
- Extracción de coordinaciones embebidas

Colapso de conjunciones múltiples Considerar el ejemplo del grupo nominal `grup.nom1234`. Es una coordinación de tres elementos: tres fechas, a pesar de que la del medio está etiquetada como número en vez de fecha. El grupo nominal tal cual está etiquetado en AnCora se muestra en la figura 4.21.

Otra particularidad que tiene este constituyente es que uno de los separadores de la coordinación es compuesto: se utiliza una coma seguida de la conjunción "y". El proceso de transformación de coordinaciones tiene una primera etapa que agrupa los separadores de conjunciones y otros elementos que aparez-

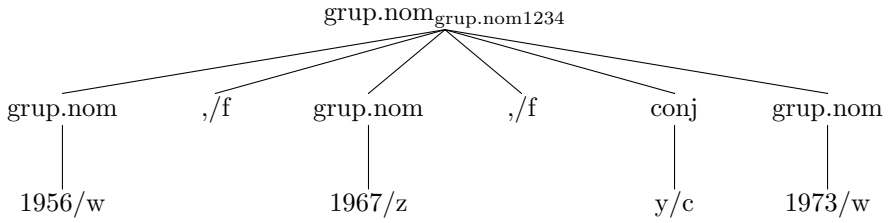

Figura 4.21: Grupo nominal `grup.nom1234` original

can que puedan introducir ruido para procesar la coordinación. A esto se le denomina *colapso de conjunciones múltiples*.

Se consideran separadores de coordinación válidos a los siguientes:

- Símbolos de puntuación coma (","), punto y coma (";") y barra ("/").
- Símbolo de puntuación punto (".") . Este es un caso más extraño, pero en AnCora aparecen casos en que varias oraciones separadas por puntos aparecen etiquetadas como una sola oración coordinada. Por ejemplo, la oración subordinada `S380`: [["] [Estoy harto de esta situación y de hablar de mi futuro] [.] [Tengo contrato hasta el 2.000 y no hay que darle más vueltas] ["]]
- Elementos de tipo `conj` marcados como conjunción coordinante (por ejemplo "y", "pero", "o").
- Elementos marcados como conjunción colapsada (o sea que haya resultado de este proceso de colapso de conjunciones múltiples).
- Elementos que se consideran separadores de coordinación excepcionales: `c5954` ("Y" mayúscula, que no está contenida en un elemento `conj`) y `c10177` ("ó" que no está contenida en un elemento `conj`).

El colapso de conjunciones múltiples agrupa una secuencia de elementos (símbolos de puntuación) con ciertas características que precedan a un separador de coordinación válido. Los diferentes casos que se agrupan son:

- Dos separadores de coordinación válidos seguidos. Por ejemplo la oración `sentence2932`: [[No voy de lista] [,] [no hago preguntas] [,] [ni] [doy soluciones] [.]]
- Comillas o símbolos de interrogación que abren ("¿") desbalanceados, seguidos de un separador de coordinación válido. Por ejemplo, la oración `sentence39`: [[De_la_Torre considera que la medalla de oro en Sydney estará situada a 2,38 metros del suelo ,] ["] [y] [Javier está capacitado para saltar esa altura " , dijo] [.]] Notar que la primera comilla está desbalanceada, porque la comilla que cierra está en otro constituyente con distinto nivel de anidamiento.

- Un guion (“-”) seguido de un separador de coordinación válido. Por ejemplo la oración subordinada `S48268`: [[se pueden fabricar] [-] [y] [se fabrican] [-]]
- Dos guiones (“-”) seguidos de un separador de coordinación válido. Por ejemplo la oración subordinada `S48984`: [[intentar] [-] [-] [y] [conseguir] [-] [-]]
- Un símbolo de interrogación que cierra (“?”), seguido de un símbolo de interrogación que abre (“¿”) seguido de un separador de coordinación válido. Por ejemplo el sintagma nominal `sn50413`: [[a un concejal , a un diputado , (...)] [?] [¿] [O] [la introducción de la figura (...)]]

Luego de este paso, el grupo nominal `grup.nom1234` es transformado y pasa a quedar como se muestra en la figura 4.22.

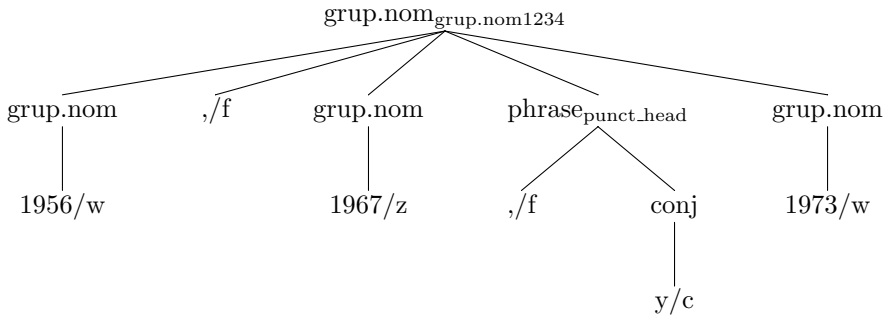

Figura 4.22: Grupo nominal `grup.nom1234` luego del colapso de conjunciones múltiples

La frase nueva que se crea tiene un atributo para indicar que es una conjunción colapsada.

Binarización de secuencias La siguiente etapa del proceso asume que un constituyente que sea una coordinación, tendrá la forma de la siguiente expresión regular:

`[elemento separador]+ elemento`

Esto significa que puede haber cualquier cantidad de elementos (por lo menos dos), siempre que todos estén separados por exactamente un separador de coordinación válido. La etapa de colapso de separadores múltiples se asegura de que, si hay más de un separador de coordinación, estos estarán colapsados en un solo constituyente, por lo que se mostrarán como uno solo para el proceso.

De esta manera, en la segunda etapa se procederá a descomponer la estructura de la coordinación múltiple para crear una cadena de aplicaciones de las reglas binarias. El resultado para el grupo nominal `grup.nom1234` es el que se muestra en la figura 4.23.

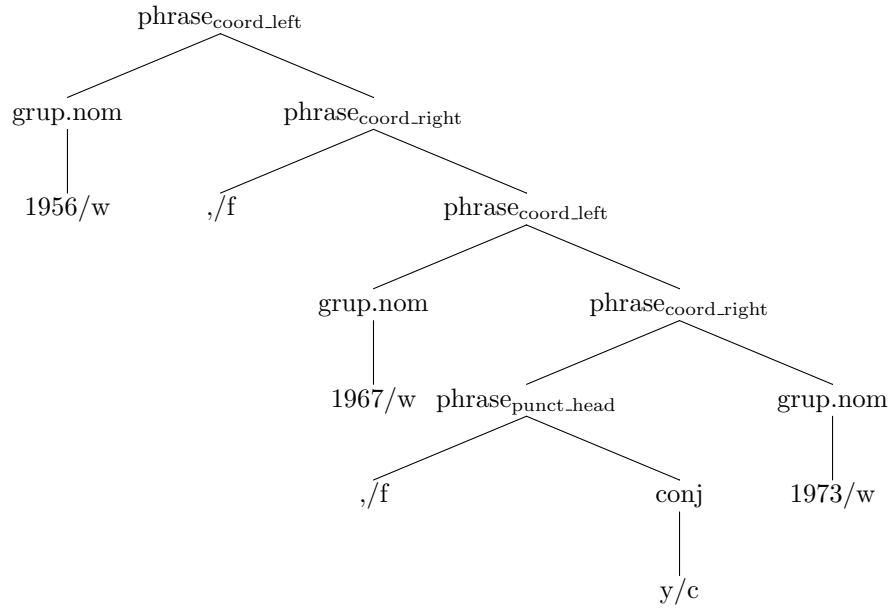

Figura 4.23: Grupo nominal `grup.nom1234` luego de la binarización de secuencias

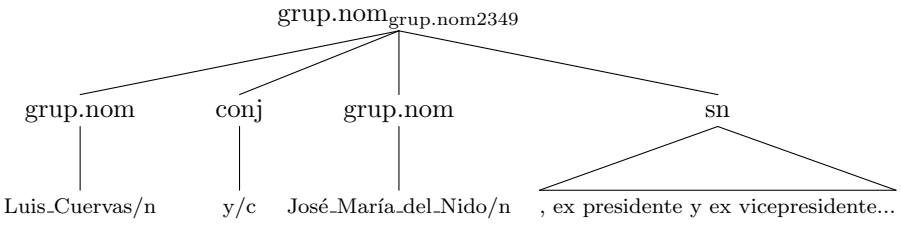

Figura 4.24: Grupo nominal `grup.nom2349` original

Extracción de coordinaciones embebidas En algunos casos las coordinaciones no se extienden en todo el constituyente, sino solamente en un aparte de él. Por ejemplo, considerar el grupo nominal `grup.nom2349` que se muestra en la figura 4.24.

Este grupo nominal contiene un grupo nominal coordinado anidado ("Luis_Cuervas y José_María_del_Nido") que aparece plano dentro del constituyente. En la etapa de *extracción de coordinaciones embebidas* se intenta identificar estos casos y extraerlos en un nodo separado, para que más adelante en el proceso *top-down* se transformen de la manera descrita en los pasos anteriores.

En este caso, el resultado de esta etapa es el que se muestra en la figura 4.25. Notar que se extrajo un nodo que quedó ternario en lugar de binario. Se espera que al continuar el proceso *top-down*, se continúe procesando ese nodo ternario y se lo binarice en la próxima etapa.

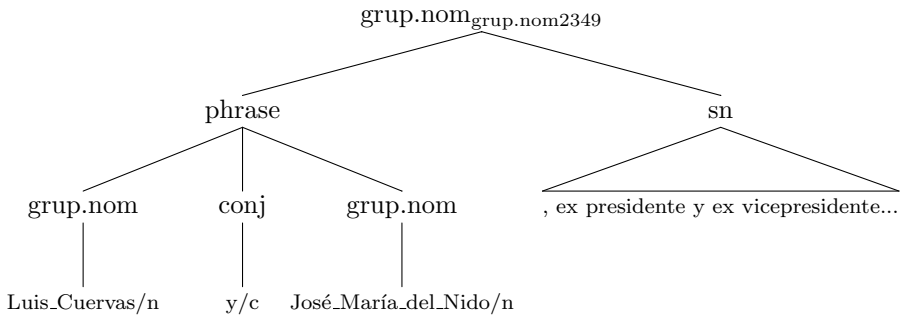

Figura 4.25: Grupo nominal `grup.nom2349` luego de la extracción de coordinaciones embebidas

4.4 Proceso bottom-up

El objetivo del proceso *top-down* es simplificar los constituyentes más complejos para llevarlos a una combinación de constituyentes más simples. Consideramos que el tipo de constituyente más simple que podemos armar en un árbol HPSG es un núcleo rodeado de sus complementos, modificadores y especificador. El proceso *top-down* intenta llevar todos los constituyentes a esta forma, mientras que el proceso *bottom-up* asume que todos los constituyentes que procesará tienen esta forma.

El proceso *bottom-up* consta de dos fases: en primer lugar se debe identificar el núcleo de un constituyente, luego se procede a clasificar cada uno de los elementos que lo rodean como especificador, complemento o modificador.

Consideremos el ejemplo del sintagma nominal `sn1`. En la figura 4.26 se muestra tal cual aparece anotado en AnCora.

Podemos ver que para construir un sintagma nominal se utilizan los constituyentes `grup.nom` y `sn`. Un `grup.nom` en general corresponde con un grupo nominal al que le falta un especificador (que es opcional) para convertirse en sintagma. Un `sn` toma un `grup.nom` y un especificador (`spec`) y genera un sintagma. En este ejemplo podemos ver tres sintagmas nominales construidos de esta manera:

- `sn1` se compone del `spec` "el" y el `grup.nom` "grupo estatal Electricité_de_France (EDF)".
- `sn2` se compone del `grup.nom` "Electricité_de_France (EDF)", y no tiene `spec`.
- `sn3` se compone del `grup.nom` "(EDF)", y tampoco tiene `spec`.

A su vez, podemos ver diferentes formas de construir un grupo nominal (`grup.nom`):

- `grup.nom1` contiene un nombre "grupo", un sintagma adjetival "estatal" y un sintagma nominal anidado "Electricité_de_France (EDF)".

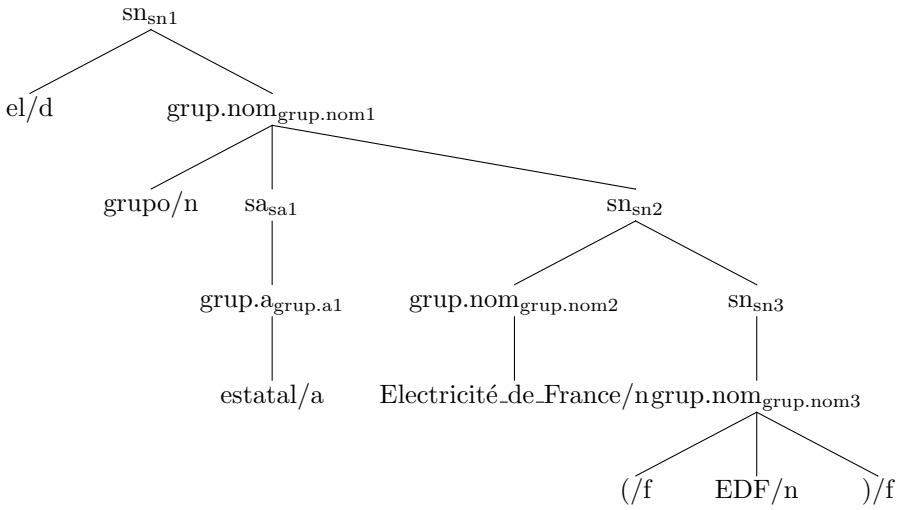

Figura 4.26: Sintagma nominal `sn1`

- `grup.nom2` contiene solamente un nombre propio "Electricité_de_France".
- `grup.nom3` contiene el símbolo de puntuación "(", seguido del nombre propio "EDF", seguido del símbolo de puntuación ")".

Pero también existen excepciones a esta estructura de sintagmas y grupos nominales en el corpus. Por ejemplo el grupo nominal `grup.nom11966` está contenido en un sintagma preposicional. Su estructura se muestra en la figura 4.27.

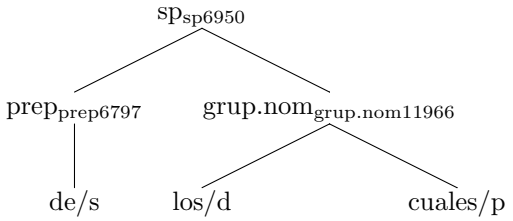

Figura 4.27: Sintagma preposicional `sp6950`

Este ejemplo tiene por lo menos dos discrepancias respecto al anterior: en primer lugar el `grup.nom` no está siendo contenido dentro de un `sn`, y en segundo lugar el propio `grup.nom` contiene el especificador (el determinante "los"), que tampoco está contenido dentro de un elemento `spec`. Por lo tanto las heurísticas de transformación deben tener en cuenta este tipo de variantes y ser robustas ante la presencia o ausencia de elementos en los distintos constituyentes.

4.4.1 Identificación de núcleos sintácticos

Para el idioma inglés, el corpus con anotaciones sintácticas más utilizado es el Penn Treebank [31]. Este corpus también está anotado al estilo de las gramáticas libres de contexto (enriquecida con atributos) y por lo tanto tampoco se indica cuál es el núcleo sintáctico de cada constituyente. En trabajos donde se ha necesitado obtener este dato (por ejemplo para gramáticas categoriales [45] o HPSG [46]) se ha recurrido a heurísticas que logran identificar el núcleo sintáctico. Para el idioma inglés, la heurística más utilizada fue desarrollada por Collins [47] para la creación de su *parser* sintáctico.

La tabla de percolación de Collins contiene reglas para identificar el núcleo sintáctico de un constituyente del Penn Treebank, pero estas reglas no son fácilmente adaptables al idioma español debido a las diferencias gramaticales entre los dos idiomas. Por ejemplo, una diferencia clara es que en inglés los adjetivos se escriben usualmente a la izquierda del sustantivo que modifican, mientras que en español se escriben a la derecha. Por lo tanto, en inglés puede resultar más natural intentar buscar el núcleo sintáctico de un grupo nominal recorriendo de derecha a izquierda, mientras que en español puede resultar más natural recorrerlo de izquierda a derecha.

Manualmente se desarrollaron reglas heurísticas que permiten identificar los núcleos sintácticos de los constituyentes de AnCora. Para cada tipo de constituyente existe una lista priorizada de reglas que permiten identificar un núcleo. Cada regla es un descriptor del tipo de estructura que se puede considerar núcleo del constituyente.

Las reglas son descriptores de constituyentes, permiten describir un elemento en una notación simple incluyendo el tipo de nodo y opcionalmente sus posibles atributos y otros predicados relevantes. Por ejemplo:

$$S/clausetype = relative$$

Es una regla que reconoce un constituyente de tipo oración subordinada, que además tiene marcado el atributo `clausetype="relative"`, lo cual significa que es de tipo relativa. La sintaxis y semántica completa de las reglas se describe en la sección 4.4.3.

Para cada constituyente de AnCora se escribió una lista de reglas para identificar su núcleo. Sea un constituyente de AnCora C de tipo t_C, este constituyente está formado por una lista de hijos:

$$C = [C1, C_2, ...C_n]$$

Sea r la lista de reglas de identificación de núcleo sintáctico para t_C:

$$r = [r_1, r_2, ...r_n]$$

El proceso de identificación de núcleos funciona como se describe en el algoritmo 1.

Esta recorrida se realiza de manera ordenada, por lo que en la práctica se establece orden de prioridad entre las reglas: si la regla r_1 reconoce algún

Algoritmo 1 Detección del núcleo de un constituyente

```
para todo r_i perteneciente a r hacer
  para todo C_i hijo del constituyente C hacer
    si r_i reconoce al elemento C_i entonces
      devolver C_i como núcleo
    fin si
  fin para
fin para
```

elemento, ese elemento es considerado como núcleo sin importar lo que la digan el resto de las reglas de menor prioridad. En cada paso, los hijos del constituyente se recorren de izquierda a derecha.

4.4.2 Clasificación de argumentos

Luego de identificar el núcleo sintáctico de un constituyente, se procede a clasificar el resto de los elementos hijos del constituyente, en función del núcleo detectado. Este proceso sigue siempre la misma lógica, descrita en el algoritmo 2[1].

El constituyente C de tipo t_C está formado por su lista de hijos $[C_1, C_2, ...C_n]$. Uno de estos hijos, C_H, está identificado como el núcleo.

De esto se desprende que para cada tipo de constituyente t_C existen tres listas de reglas que permiten clasificar los elementos que acompañan el núcleo:

- Reglas de detección de complemento
- Reglas de detección de modificador
- Reglas de detección de especificador

Las reglas en estas listas son análogas a las reglas utilizadas para la identificación del núcleo, y están escritas en el mismo lenguaje. Existen además dos listas de reglas de detección de símbolos de puntuación (una para los símbolos que abren y otra para los símbolos que cierran). Estas reglas no dependen del tipo de constituyente.

[1] Este algoritmo describe la identificación de los complementos, modificadores, especificador y puntuación. En la transformación *bottom-up* de las oraciones se utiliza además una etapa extra para la identificación de clíticos.

Algoritmo 2 Clasificación de argumentos

mientras no quede solamente C_H en la lista **hacer**
  mediante la lista de reglas para detectar complementos de t_C, detectar si C_{H-1} o C_{H+1} (elementos a la izquierda o a la derecha del núcleo) es complemento
  si uno de ellos es complemento **entonces**
    llamamos *Comp* al nodo detectado, construir un nuevo nodo con regla del complemento agrupando C_H y *Comp*, este es el nuevo nodo C_H
  si no
    mediante la lista de reglas para detectar modificadores de t_C, detectar si C_{H-1} o C_{H+1} es modificador
    si uno de ellos es modificador **entonces**
      llamamos *Mod* al nodo detectado, construir un nuevo nodo con regla del modificador agrupando C_H y *Mod*, este es el nuevo nodo C_H
    si no
      mediante la lista de reglas para detectar especificadores de t_C, detectar si C_{H-1} o C_{H+1} es especificador
      si uno de ellos es especificador **entonces**
        llamamos *Spec* al nodo detectado, construir un nuevo nodo con regla del especificador agrupando C_H y *Spec*, este es el nuevo nodo C_H
      si no
        mediante la lista de reglas para detectar puntuación, detectar si C_{H-1} o C_{H+1} es símbolo de puntuación
        si uno de ellos es puntuación **entonces**
          llamamos *Punct* al nodo detectado, construir un nuevo nodo con regla de puntuación agrupando C_H y *Punct*, este es el nuevo nodo C_H
        si no
          fallar (significa que el constituyente no era del tipo de árbol simple que esperábamos)
        fin si
      fin si
    fin si
  fin si
fin mientras

4.4.3 Reglas de identificación de elementos

La figura 4.28 muestra la sintaxis de las reglas de identificación de elementos en formato BNF[2] que se utilizan para la identificación tanto de núcleos como de argumentos.

Decimos que un elemento es un *match* para una regla si cumple con lo especificado por la regla. Las condiciones para que un elemento sea un *match* se describen a continuación.

[2]Se utiliza la notación BNF extendida con el uso de [ ] para denotar elementos opcionales.

```
      <rule> ::= <category> ["/" <attributes>]
               |  "!" <category>
  <category> ::= "*"
               |  <ancora_category>
<attributes> ::= <attribute> ["^" <attributes>]
 <attribute> ::= ["!"] <attr_name> ["=" <attr_value>]
 <attr_name> ::= "yield"
               |  "child"
               |  "child.wd"
               |  "originalId"
               |  <string>
<attr_value> ::= <string>
```

Figura 4.28: Sintaxis de las reglas de identificación de elementos

Una regla siempre define una categoría de AnCora a identificar (por ejemplo `grup.nom`, `S` o `n`) y opcionalmente una lista de atributos. La categoría puede estar negada, lo cual indica que se producirá un *match* con cualquier elemento excepto los que tengan la categoría indicada. Existe un símbolo especial (`'*'`) para indicar un *match* con cualquier categoría.

La sección de atributos es una lista de descriptores de atributos separados por un símbolo `'^'`. Un descriptor de atributos siempre indica un nombre de atributo y opcionalmente puede especificar un valor y puede estar negado.

- Si solamente aparece el nombre del atributo, se espera que un *match* contenga ese atributo sin importar su valor.
- Si se especifica el valor del atributo, se espera que un *match* contenga el atributo con el mismo valor.
- Si el descriptor está negado, se espera que o bien no esté el atributo, o bien esté definido con un valor diferente del especificado (en caso de que exista).

Si una regla contiene varios descriptores de atributos separados por `'^'`, un *match* debe cumplir con todos los descriptores.

Se puede utilizar cualquier string como nombre de atributo, pero existen algunos nombres reservados que se utilizan para definir comportamientos particulares:

- `yield`: Para que un elemento sea *match*, se debe cumplir que el *yield* de este elemento (o sea la secuencia de palabras del subárbol correspondiente al elemento) tenga el mismo valor que el especificado en `<attr_value>`.
- `child`: Para que un elemento sea *match*, debe tener un solo elemento hijo y su categoría debe ser igual a la indicada por el valor de `<attr_value>`.

- `child.wd`: Para que un elemento sea *match*, debe tener un solo elemento hijo, el cual debe tener un atributo `wd` y su valor debe ser igual al indicado por el valor de `<attr_value>`.

- `originalId`: A medida que avanza el proceso de conversión, muchos elementos son movidos o sustituidos por otros nuevos, por lo que su identificador original se pierde y pasan a tener uno nuevo. Utilizando el atributo `originalId` se intenta busca el identificador original del elemento antes de la transformación. Un *match* es un elemento que en el corpus original tenía el identificador `<attr_value>`.

4.4.4 Grupos nominales

Los grupos nominales (`grup.nom`) y los sintagmas nominales (`sn`) se transforman por separado, debido a que tienen diferentes reglas. En los ejemplos que se describieron anteriormente se muestran algunas de las estructuras que es posible encontrar como núcleo de un grupo nominal:

- Los núcleos sintácticos de `grup.nom1`, `grup.nom2` y `grup.nom3` son nombres ("grupo", "Electricité_de_France" y "EDF", respectivamente).

- El núcleo sintáctico de `grup.nom11966` es "cuales", un pronombre.

Es posible encontrar otros tipos de elementos como núcleos de grupos nominales. La tabla 4.3 muestra el conjunto de reglas que se utilizan para detectar el núcleo sintáctico de un grupo nominal de AnCora, junto con un ejemplo de qué caso está contemplando. Algunos de los casos se deben a ejemplos ruidosos: mediante la regla v se identifica como núcleo a la palabra "capitulo", que está marcada como verbo en el corpus. Esto es un error en el corpus, la palabra correcta debió ser "capítulo" y estar marcada como nombre.

La tabla 4.4 muestra algunas de las reglas utilizadas para clasificar un elemento que acompaña al núcleo como complemento en el contexto de un grupo nominal (la tabla completa de las reglas puede verse en el anexo A.1). En los ejemplos, se marca como `H` el núcleo sintáctico y como `C` el complemento identificado por la regla.

La tabla 4.5 muestra algunas de las reglas que se aplican para la detección de modificadores en el contexto de un grupo nominal (la tabla completa de las reglas puede verse en el anexo A.2). En los ejemplos, se marca como `H` el núcleo sintáctico y como `M` el modificador identificado por la regla.

La tabla 4.6 muestra las reglas que se aplican para la detección de especificadores en el contexto de un grupo nominal. En los ejemplos, se marca como `H` el núcleo sintáctico y como `S` el especificador identificado por la regla.

- `n` (sustantivo)
  "... Río_Bravo y Saltillo para la [[$_H$ compañía] [francesa]]..."
- `grup.nom` (grupo nominal anidado)
  "... y sobre [[$_H$ transmisiones y retenciones] [de fondos de inversión]] ."
- `p` (pronombre)
  "... obtuvo 19 diputados, [[$_H$ dos] [más]] que en 1996..."
- `w` (fecha)
  "... hundimiento del "Kursk" el [[pasado] [$_H$ 12_de_agosto]] en aguas árticas..."
- `z` (número)
  "... donde lograron el [[$_H$ 71_por_ciento] [de los sufragios]] ..."
- `a` (adjetivo)
  "... quien cuestiona al entrenador es [[$_H$ enemigo] [del Barça]] ."
- `v` (verbo)
  "... sobre todo en el [[$_H$ capitulo] [de las infraestructuras]] ..."
- `s.a` (sintagma adjetival)
  "... y la [[$_H$ segunda] [, mucho más potente,]] a las 07.30.42..."
- `participi` (participio)
  "... el relato ZZadjNM de lo [[$_H$ ocurrido] [en la sima de ZZlugar]] ..."
- `S/clausetype=participle` (oración subordinada de tipo participio)
  "... en_lugar_del [[$_H$ destituido] [Carlos_Sainz_de_Aja]] ."
- `S/clausetype=relative` (oración subordinada de tipo relativa)
  "... incluidos los [[$_H$ que él mismo ha hablado] [sobre sí mismo]] ..."
- `S/clausetype=completive` (oración subordinada de tipo completiva)
  "Al [[$_H$ correr] [de los siglos]] se había manifestado un..."
- `sp` (sintagma preposicional)
  "aeropuerto de Miami, uno de los [[$_H$ de mayor tráfico aéreo] [en EEUU]]..."
- `sn` (sintagma nominal)
  "... el hotel (un [[$_H$ cinco estrellas de gran lujo]])..."

Tabla 4.3: Reglas para identificación de núcleo de un `grup.nom`

- `sp/arg=arg0∧!adjunct=yes`
  `grup.nom31`: [[$_H$ acuerdo] [de venta de energía] [$_C$ de EAA] [con la Comisión_Federal_de_Electricidad (CFE)]]
- `sp/arg=arg1∧!adjunct=yes`
  `grup.nom5`: [[$_H$ compra] [$_C$ del 51_por_ciento de la empresa mexicana (...)]]
- `sp/func=cn∧arg=arg0`
  `grup.nom10428`: [[$_H$ gestos y pasos "] [$_C$ del PP]]
- `sp/func=cn∧arg=arg1`
  `grup.nom11272`: [[$_H$ ampliación y modernización] [$_C$ de la Autopista_del_Oeste (...)]]

Tabla 4.4: Algunas reglas para identificación de complementos de un `grup.nom`

– `sp/!arg`
  `grup.nom24`: [[H combustible] [principal] [M en una central de ciclo combinado (...)]]
– `s.a`
  `grup.nom56173`: [["] [H abuso] ["] [M empresarial] [en la utilización de ese tipo de contratación]]
– `a`
  `grup.nom5824`: [[M ex] [H vicepresidente]]
– `S`
  `grup.nom56413`: [[H grupo] [M presidido por Ballvé]]
– `sadv`
  `grup.nom37482`: [[H minutos] [M más] [que se jugaron en el estadio San_Carlos_de_Apoquindo , en Santiago]]
– `neg`
  `grup.nom52398`: [[M no] [H intervención]]

Tabla 4.5: Algunas reglas para identificación de modificadores de un `grup.nom`

– `spec`
  `grup.nom560`: [[S los] [H cuales]]
– `d`
  `grup.nom3368`: [[S una] [H de las mejores virtudes del filme]]
– `z`
  `grup.nom47367`: [[S 15] [H millones] [de ordenadores que intentaron entrar en (. . .)]]

Tabla 4.6: Reglas para identificación del especificador de un `grup.nom`

4.4.5 Sintagmas nominales

La tabla 4.7 muestra las reglas utilizadas para reconocer el núcleo sintáctico de los sintagmas nominales (`sn`), junto con un ejemplo de los casos que está contemplando. Los primeros dos casos son excepciones debidas a anotaciones erróneas en el corpus.

- `sadv/id=sn49611`
  "... sentenciada en rebeldía , [[también] [$_{H}$hace dos días]] , a cinco años ." Esta regla es una excepción que trata de identificar el núcleo correcto a pesar de que el constituyente entero está mal etiquetado en AnCora: "también hace dos días" no debería ser un sintagma nominal.
- `grup.nom/id=grup.nom7820`
  "... ha valido a Burlotti sumar [[40] [$_{H}$puntos]] , mientras_que ..." En esta excepción el determinante "40" está erróneamente identificado como un `grup.nom` en AnCora, por lo que se marca explícitamente el `grup.nom` "puntos" como núcleo.
- `grup.nom`
  "... creada por [[el] [$_{H}$japonés Mitsubishi_Corporation]] para poner_en_marcha ..."
- `sn`
  "... en la que [[$_{H}$los luchadores jugadores brasileños] [, salvo el " todoterreno " Gustavo_Kuerten ,]] no se adaptan bien ..."
- `relatiu`
  "... la pequeña y mediana empresa , Pimec-Sefes , en [[la] [$_{H}$que]] también advirtió a los empresarios ..."
- `S/clausetype=relative`
  "... establezca su visión , [[lo] [$_{H}$cual prolongaría el proceso judicial muchos meses]] ."
- `s.a`
  "... algunas no han podido y otras [[,] [las] [$_{H}$más importantes] [,]] no han sabido ."

Tabla 4.7: Reglas para identificación de núcleo de un `sn`

4.4.6 Sintagmas y grupos adjetivales

En AnCora existen tres tipos de no terminales relacionados con los adjetivos: `grup.a`, `s.a` y `sa`. La diferencia entre estas tres estructuras no es clara. Según la documentación de AnCora [48], estas estructuras representan:

- `grup.a`: Grupo adjetival
- `s.a`: Frase adjetival
- `sa`: Frase adjetival (dependiendo de una estructura causal)

En el caso de los nombres, la diferencia entre un grupo nominal y un sintagma nominal puede considerarse que está en la ausencia o presencia del especificador del sintagma. Sin embargo, esta diferencia no existiría en el caso de los adjetivos, por lo que en la práctica es difícil establecer una diferencia entre grupo y frase adjetival.

Analizando las apariciones de los constituyentes en el corpus, puede verse el siguiente patrón:

- Los `grup.a` suelen estar anidados en un `sa`, en un `s.a` o en otro `grup.a`, aunque también pueden aparecer solos. Por ejemplo: los grupos adjetivales `grup.a13804` ("estadounidense") y `grup.a13805` ("procedente de la base de Incirlik , en el sur de Turquía ,") que pertenece al `grup.nom65566` [[caza] [F-16] [estadounidense] [procedente de la base de Incirlik , en el sur de Turquía ,]]

- Además del `grup.a`, un `sa` o un `s.a` pueden incluir un elemento `spec`, que en general no es un especificador sino un modificador (son adverbios, por ejemplo en `s.a10036` "[[ya] [existentes]]"). También pueden incluir una negación u otros elementos que modifican al núcleo.

Estas reglas no son estrictas, y en general los tres constituyentes parecen tener el mismo tipo de contenido.

Para la identificación de núcleo de los `grup.a`, se utilizan las reglas indicadas en la tabla 4.8.

- `a`
  "... se mostró [[$_{\text{H}}$contrario] [a la operación de rescate de los cadáveres]] y abogó..."
- `n`
  "... será proclamado el primer judío [[$_{\text{H}}$candidato] [a la vicepresidencia de los Estados_Unidos]] ."
- `s.a`
  "... entre socialdemócratas y [[$_{\text{H}}$verdes]] , que apoyan el aislamiento de Viena..."
- `grup.a`
  "... al que califica de [["] [$_{\text{H}}$trabajador y profesional] [como pocos] ["]] ."
- `d`
  "... si el portero francés recuperará hoy [[$_{\text{H}}$mismo]] la titularidad..."
- `p`
  "... los números [[$_{\text{H}}$uno]] de cada equipo..."
- `z`
  "... la casilla 58 [[$_{\text{H}}$de]] la declaración de Renta..."
- `w`
  "... la clase de Religión en el curso [[$_{\text{H}}$1997-98]] se ha pasado a..."
- `S`
  "... está [[$_{\text{H}}$previsto]] el comienzo de una operación para rescatar..."
- `neg`
  "... a la seducción [[$_{\text{H}}$no]] ruda ."

Tabla 4.8: Reglas para identificación de núcleo de un `grup.a`

Para la identificación de núcleos de `s.a` y `sa` se utiliza el mismo conjunto de reglas, que se muestra en la tabla 4.9.

– `n/id=n1386` "...sometidas a retención o ingreso a de indemnizaciones y [[Hsubvenciones] [agrícolas]] ; sobre pagos fraccionados..." Este es un caso excepcional, ya que "subvenciones agrícolas" está etiquetado por error como sintagma adjetival, cuando debería ser un sintagma nominal.
– `grup.a` "...aunque en este municipio por [[muy] [Hescaso]] margen de votos..."
– `a` "...el rival " tiene [[Hdelanteros] [muy peligrosos]] " ."
– `sa` "..." ninguna de las tres versiones es [[Hmás verosímil] [que la otra]] " , aunque así..."
– `s.a` "...y sus dimensiones , [[Hmucho menores] [que Rover]] , aunque no precisó..."

Tabla 4.9: Reglas para identificación de núcleo de un `s.a`

4.4.7 Grupos verbales

En Acora, los grupos verbales (elementos `grup.verb`) se utilizan para representar verbos simples, verbos con sus auxiliares o perífrasis verbales. Estos son algunos ejemplos:

- `grup.verb10`: [[participaron]]
- `grup.verb10005`: [[debe] [mantener]]
- `grup.verb10028`: [[tienen] [que] [hacer]]
- `grup.verb10365`: [[vamos] [a] [comenzar a volar]]
- `grup.verb11903`: [[ha] [guiado]]

Algunas veces se incluyen dentro del `grup.verb` otras estructuras que no son estrictamente verbales. Por ejemplo puede aparecer el sujeto dentro del `grup.verb`:

- `grup.verb38886`: [[tendrá] [Romario] [que] [ir]]

Estos constituyentes representan un problema respecto al paradigma de procesamiento con el que tratamos al resto de los constituyentes. Considerar el ejemplo de `grup.verb10028` que se muestra en la figura 4.29.

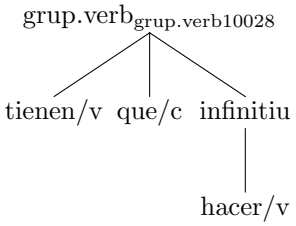

Figura 4.29: Grupo verbal `grup.verb38886` original

Mediante reglas podemos detectar que el núcleo es el primer verbo ("tienen"), y desde el punto de vista sintáctico esto sería correcto, ya que el sujeto que

acompañe a este grupo verbal deberá concordar en número y persona con este verbo. Sin embargo, no es posible discriminar cuál es el rol de la conjunción "que" y del verbo infinitivo "hacer". Se necesita conocimiento extra para interpretar que "que hacer" deben formar una unidad, y esa unidad es complemento de "tienen". Utilizar el enfoque de lista de reglas que se usa para los otros constituyentes no permite transformar bien estos casos.

Considerar además la oración subordinada donde `S12021`, donde se usa este grupo verbal, como se muestra en la figura 4.30.

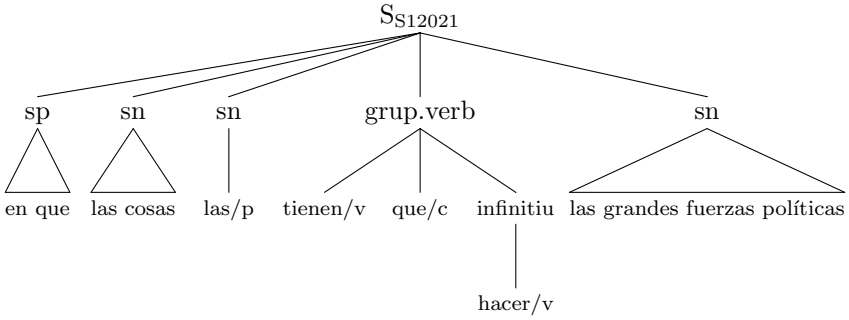

Figura 4.30: Oración subordinada `S12021` original

En primer lugar, esta oración comienza con una expresión pronominal relativa, por lo que debe haber sido transformada en el proceso *top-down* para quedar como se muestra en la figura 4.31.

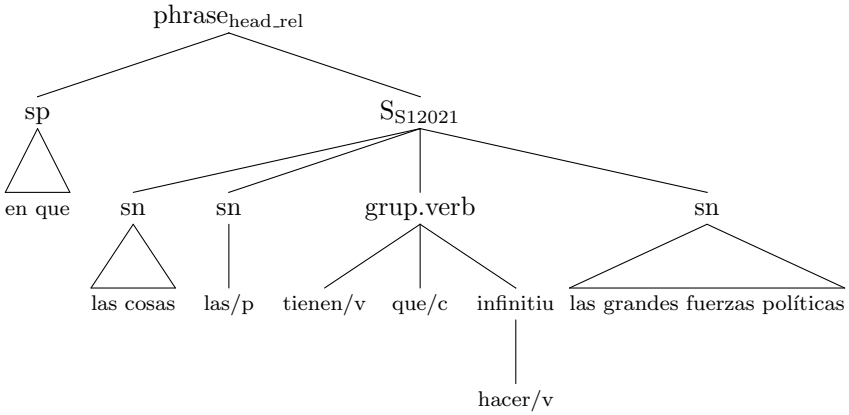

Figura 4.31: Oración subordinada `S12021` luego de la extracción de relativa

En este ejemplo "las grandes fuerzas políticas" es el sujeto, y "las cosas" es el complemento directo. El sujeto puede asignarse tanto a "tienen" como a "hacer", sin embargo el complemento directo solo puede asignarse a "hacer". Si elegimos "tienen" como núcleo, ambos parámetros quedarían asociados a este verbo, lo cual sería un error. Lo que ocurre es que en estos casos existe un

núcleo sintáctico ("tienen") con el que el sujeto debe concordar, y otro núcleo semántico ("hacer") que es el que elige los argumentos que la oración debe tener.

Para modelar esto, la estructura resultante del grupo verbal debe contener la información necesaria para identificar tanto el núcleo sintáctico como el núcleo semántico.

Para detectar los núcleos sintáctico y semántico del grupo verbal, se parte de la estructura plana del `grup.verb` y se aplican una serie de heurísticas que lo binarizan. Las heurísticas operan de la siguiente manera: sea $C = [C_1, C_2, \ldots C_n]$ la lista de hijos del grupo verbal, y $p = [p_1, p_2, \ldots p_m]$ una lista de patrones (los patrones, como veremos más adelante, son secuencias de reglas), para transformar el grupo verbal se utiliza el algoritmo 3.

Algoritmo 3 Transformación de un grupo verbal

mientras haya más de un elemento en C **hacer**
  para todo p_i en p **hacer**
    Sea L_p el largo de p_i
    para todo C_s subsecuencia de C con L_p elementos *(comenzando por la de más a la derecha, o sea* $[C_{n-L_p+1}, \ldots C_n]$*)* **hacer**
      si C_s cumple con el patrón p_i **entonces**
        Se elige dos nodos de C_s *(cuáles son depende del caso)*
        Se crea un nuevo nodo C_i que agrupa esos nodos *(la regla a aplicar, el nodo a seleccionar como núcleo y el nodo a seleccionar como núcleo semántico dependen del caso)*
        Sustituir los nodos en C por el nodo C_i
      fin si
    fin para
  fin para
fin mientras

Cada uno de los patrones utilizados define una secuencia de restricciones que tienen que cumplir los elementos, y además selecciona cómo va a ser el nuevo nodo que se construye y qué núcleos va a tener. La tabla 4.10 muestra cuáles son los patrones que se usan.

En el ejemplo de `grup.verb10028` primero se aplica la heurística de la fila cuatro, y al resultado se le aplica la heurística de la última fila. El resultado es el que se muestra en la figura 4.32, donde el núcleo sintáctico pasa a ser "tienen" y el núcleo semántico pasa a ser "hacer".

Patrón	Nuevo nodo	Regla	Núcleo sintáctico	Núcleo semántico	Ejemplo
1.[“f”] 2.[“*”]	[1, 2]	punct_head	2	2	`infinitiu420`: [[estar] [”] [usando]]
1.[“*”] 2.[“f”]	[1, 2]	head_punct	1	1	`grup.verb41294`: [[podía] [comer] [;]]
1.[“s”,“prep”] 2.[“*”]	[1, 2]	head_comp	1	2	`grup.verb41561`: [[vuelve] [a] [hacer]]
1.[“c”] 2.[“*”]	[1, 2]	head_comp	1	2	`grup.verb41623`: [[hay] [que] [tener]]
1.[“v”, “grup.verb”] 2.[“sn/tem=tmp”]	[1, 2]	head_comp	1	1	`grup.verb2959`: [[llevan] [ya] [varios días] [ofreciendo]]
1.[“v”] 2.[“sn/yield=nada|algo|mucho”]	[1, 2]	head_mod	1	1	`grup.verb4611`: [[tiene] [nada] [que] [ver]]
1.[“v”] 2.[“sn/func=suj”]	[1, 2]	head_spec	1	1	`grup.verb12275`: [[puede] [un informe sobre (. . .)] [ser]]
1.[“v”] 2.[“sn/func=cc”]	[1, 2]	head_mod	1	1	`grup.verb39120`: [[tienen] [muy poco] [que] [ver]]
1.[“v”] 2.[“sn”]	[1, 2]	head_comp	1	1	`grup.verb4280`: [[hizo] [especial] [hincapié]*]
1.[“v”] 2.[“sadv/tem=tmp”, “sadv/id=sadv5458|sadv5858 |sadv8021|sadv8325|sadv8326 |sadv8819|sadv11443|sadv12171”] 3. [“*”]	[1, 2]	head_mod	1	1	`grup.verb6663`: [[hubiera] [nunca] [sufrido]]
1.[“v/lem=ser”] 2.[“sadv”] 3.[“*”]	[2, 3]	mod_head	3	3	`infinitiu2240`: [[ser] [bien] [acogido]]
1.[“neg”] 2.[“infinitiu”]	[1, 2]	mod_head	2	2	`grup.verb17828`: [[podrían] [no] [renovar]]
1.[“v”,“grup.verb”] 2.[“sadv”]	[1, 2]	head_mod	1	1	`grup.verb18365`: [[debería] [también] [poner]]
1.[“s.a”] 2.[“n”]	[1, 2]	mod_head	2	2	`grup.verb4280`: [[hizo] [especial] [hincapié]]
1.[“v”,“infinitiu”,“grup.verb”,“a”] 2.[“v”,“gerundi”,“infinitiu”, “participi”,“grup.verb”,“sp”, “subordinate”,“n”,“s/id=s79118”]	[1, 2]	head_comp	1	2	`grup.verb4281`: [[ha] [renovado]]

* En este caso [especial hincapié] ya fue procesado por otra regla para `grup.verb4280`

Tabla 4.10: Reglas para transformación de un grupo verbal

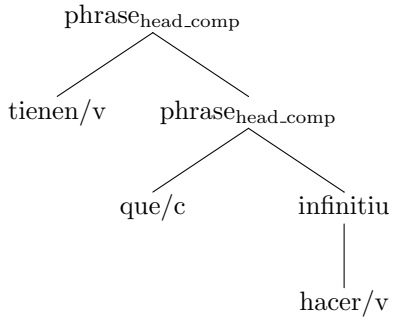

Figura 4.32: Grupo verbal `grup.verb10028` luego de la transformación *bottom-up*

4.4.8 Oraciones

Los tipos de constituyentes con mayor complejidad del corpus son los de oraciones (`sentence`) y oraciones subordinadas (`S`). Ambos se comportan de manera similar en cuanto al núcleo, ya que tienden a tener un núcleo verbal. Además se comportan de manera similar en cuanto a los argumentos, pero en el caso de las `S` es frecuente que aparezca una expresión pronominal relativa al inicio.

Por ejemplo, la oración subordinada `S68` que se muestra en la figura 4.33 comienza con una expresión pronominal relativa, mientras que la oración subordinada `S30` que se muestra en la figura 4.34 no la contiene.

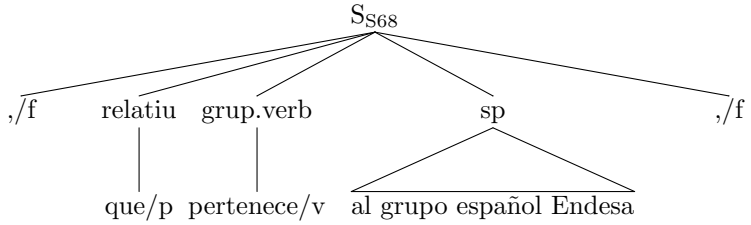

Figura 4.33: Oración subordinada `S68`

Las oraciones tipo `sentence` se comportan como esta segunda categoría, por ejemplo la oración `sentence174` que se muestra en la figura 4.35.

Sería un error considerar la expresión pronominal relativa como un argumento más, ya que no es un elemento que dependa del núcleo como los complementos, modificadores y especificador. Más bien se considera que la expresión pronominal relativa está dominando sobre toda la oración que contiene al núcleo. Debido a esto, durante el procesamiento *top-down* existe un paso que se encarga de extraer la expresión pronominal relativa (ver sección 4.3.7), la cual queda como un nodo separado del resto de la oración subordinada, el resto de los elementos se comportarán de manera análoga a una oración normal. El proceso *bottom-up* asume que estas expresiones pronominales relativas ya fueron sepa-

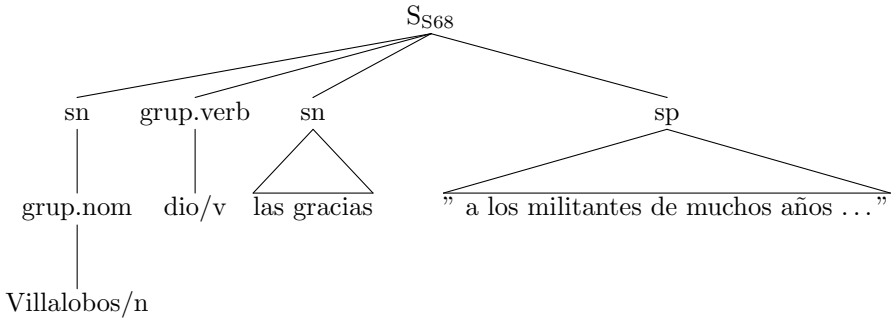

Figura 4.34: Oración subordinada `S30`

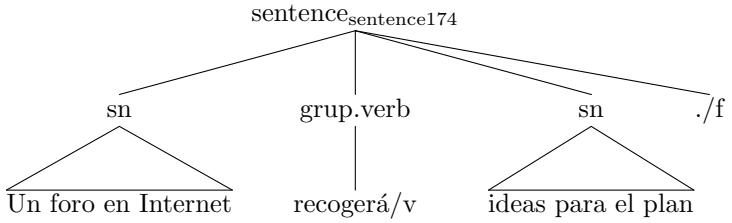

Figura 4.35: Oración `sentence174`

radas, y lo que queda dentro de la oración subordinada tiene una estructura similar a la de una oración de tipo `sentence`.

Por lo tanto se utilizan los mismos conjuntos de reglas de identificación de núcleo sintáctico y de clasificación de argumentos para `sentence` y para `S`. Algunas de las reglas para identificar el núcleo sintáctico se muestran en la tabla 4.11 (la tabla completa de reglas puede verse en el anexo A.3).

- `grup.verb`
  `sentence1`: [[El grupo estatal Electricité_de_France (EDF)] [$_{H}$anunció] [hoy , jueves ,] [la compra del 51_por_ciento de la empresa mexicana (...)] [.]]
- `infinitiu`
  `S37`: [[$_{H}$llevar] [al PP] [hasta la victoria]]
- `gerundi`
  `S421`: [[$_{H}$llevando] [en la mano] [un balón hinchable que representaba (. . .)]]
- `v`
  `S2499`: [[$_{H}$comprada] [por Entergy_Londres]]
- `S`
  `S40`: [[$_{H}$ilusionar y convencer] [a muchos] [para que apuesten " por este partido]]
- `sn`
  `S210`: [[$_{H}$la segunda , mucho más potente ,] [a las 07.30.42 GMT]]
- `sa`
  `S4409`: [[( ] [$_{H}$segundo] [ayer] [, con 8,33] [ )]]

Tabla 4.11: Algunas reglas para identificación de núcleo de una oración

Además de las tablas de reglas para complementos, modificador y especificador, en el caso de las oraciones aparece un nuevo tipo de elementos que hay que tener en cuenta: los clíticos. Los clíticos introducen una complejidad extra al momento de identificar los argumentos del verbo principal de una oración, debido a que en español existe el fenómeno de redundancia pronominal o duplicación de clíticos [49]: un clítico puede estar ocupando el lugar de un argumento (por ejemplo el complemento directo) pero además es usual que aunque el argumento exista en la oración, también se agregue el clítico. Por ejemplo: "Le di un regalo a Juan".

En AnCora los clíticos están marcados con la función gramatical del argumento que representan. Por ejemplo, observar la oración `sentence176` que se muestra en la figura 4.36.

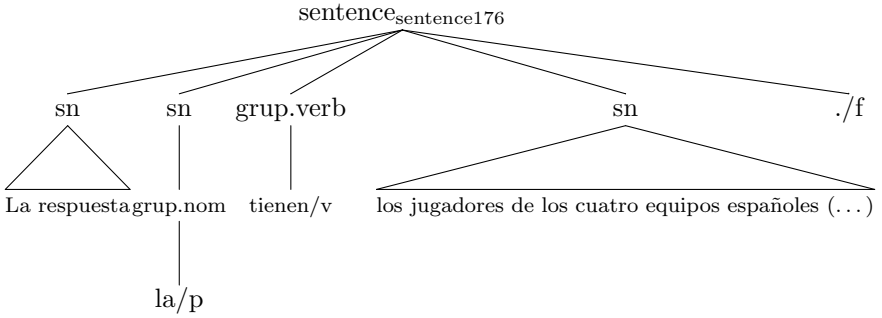

Figura 4.36: Oración `sentence176`

En este caso aparecen tanto el complemento directo "La respuesta" como el clítico del complemento directo "la". Ambos aparecen marcados en AnCora con el atributo `func='cd'`. Esto presenta dos dificultades:

- Es necesario distinguir los casos en los que el complemento directo está duplicado debido a la presencia de un clítico.
- En los casos en que solamente está el clítico, y no el verdadero complemento, se necesita lógica extra para inferir la categoría gramatical del argumento.

Para evitar esta complejidad, se tomó la decisión de marcar los clíticos de manera diferente a los argumentos propios (especificador y complementos). Como se menciona en la sección 3.3.5.2, se utiliza una regla distinta (`clitic_head`) para indicar la combinación del núcleo verbal con un clítico.

Para identificar los clíticos se hace uso del atributo marcado durante la etapa *top-down* que se describió en la sección 4.3.1. Todo elemento marcado como pronombre de tipo `morfema` o `clitico` se agrupará con el núcleo verbal mediante la regla `clitic_head`.

La tabla 4.12 muestra algunas de las reglas para clasificar un elemento que acompaña el núcleo como complemento (la tabla completa puede verse en el anexo A.4).

- `sn/func=cd` — `S2`: [[$_{H}$poner_en_marcha] [$_{C}$una central de gas de 495 megavatios]]
- `sn/func=atr` — `S121`: [[una de bronce] [no] [$_{H}$sería] [para mí] [$_{C}$una decepción]]
- `sn/func=cpred` — `S157`: [[$_{H}$denominado] [$_{C}$" cinturón rojo "]]
- `sn/func=ci` — `sentence5886`: [[De esta manera ,] [$_{C}$los inversores que quieran comprar dólares] [le] [$_{H}$costaba] [cada billete verde] [168,28 pesetas , el nivel más alto desde 1985] [.]]
- `sn/func=cc^arg=arg2` — `S20451`: [[mientras_que] [la cifra de negocios] [$_{H}$aumentó] [$_{C}$un ocho_por_ciento] [, hasta los 44.350 millones de pesetas (...)]]
- `S/func=cd` — `sentence6542`: [[Por eso ,] [$_{H}$opinó] [$_{C}$que las listas de espera no son una " enfermedad " del sistema sanitario (...)]]
- `sp/func=creg` — `S21067`: [[que] [Andalucía] [$_{H}$converja] [$_{C}$con Europa]]
- `sp/func=cag` — `S21087`: [[$_{H}$firmado] [$_{C}$por el PA] [con UAGA] [- que ya lo hizo antes con el PSOE e IU -]]
- `sp/arg=arg2` — `sentence6706`: [[La Ertzaintza , que acudió minutos después ,] [$_{H}$recogió] [$_{C}$del suelo] [casquillos del calibre 7,65 utilizados en los fusiles Cetme y de 9 milímetros] [.]]
- `sadv/func=cd` — `sentence7184`: [[-] [-] [Una imagen] [$_{H}$vale] [$_{C}$más que mil palabras] [.]]
- `sadv/func=cc^arg=arg2` — `S23298`: [[que] [su entrenador] [$_{H}$pudiera estar] [$_{C}$cerca de ella] [durante los combates]]

Tabla 4.12: Algunas reglas para identificación de complementos de una oración

La tabla 4.13 muestra las reglas para clasificar un elemento que acompaña el núcleo de una oración como especificador (sujeto).

- `sn/func=suj`
  `S12876`: [[$_{S}$La Comisión_Europea] [$_{H}$invita] [a las terceras partes interesadas en esta operación] [a trasmitirle sus observaciones (...)] [.]]
- `S/func=suj`
  `S953`: [[que] [$_{S}$quien cuestiona al entrenador] [$_{H}$es] [enemigo del Barça]]
- `sadv/func=suj`
  `S45784`: [[que] [$_{S}$más de uno] [$_{H}$esté temblando] [sólo de pensar que el martes se despierte]]

Tabla 4.13: Reglas para identificación del especificador de una oración

4.5 Evaluación

Las reglas construidas permitieron transformar la casi totalidad del corpus AnCora. En el corpus existen 780950 constituyentes, y las heurísticas desarrolladas logran un cubrimiento casi total de los mismos, dejando solamente 95 de estos constituyentes que no están cubiertos por ninguna regla [50].

La cantidad total de reglas construidas manualmente para detectar el núcleo y los diferentes tipos de argumentos puede verse en la tabla 4.14.

Se realizó una evaluación del resultado de la transformación de la siguiente manera: se tomó una muestra aleatoria de 40 oraciones (en total 779 constituyentes sintácticos) y manualmente se etiquetó:

Constituyentes	Núcleo	Complemento	Modificador	Especificador
`grup.a`	10		1	
`grup.adv`	4	1	12	
`grup.nom`	14	9	18	3
`prep`	1		8	
`sentence`, `S`	14	14	48	
`s.a`, `sa`	5		10	
`sadv`	3		12	1
`sn`	7	5	17	6
`sp`	3	14	5	
`spec`	9			

Tabla 4.14: Cantidades de reglas utilizadas para detectar núcleo, complementos, modificadores y especificador para cada tipo de constituyente

- El núcleo de cada constituyente.
- Para todos los elementos que no son el núcleo, cuál es su relación respecto al núcleo dentro de estas posibilidades: especificador, complemento, modificador, clítico o puntuación.

4.5.1 Identificación de núcleos

Se encontró que en el 95.3% de los casos el núcleo etiquetado manualmente se corresponde con el núcleo encontrado por la heurística. La tabla 4.15 muestra la precisión de las reglas de detección de núcleo agrupando los ejemplos según tipo de constituyente.

Categoría	Total	Correctos	Precisión
grup.a	9	6	66.7%
grup.adv	3	3	100.0%
grup.nom	162	154	95.1%
grup.verb	23	23	100.0%
infinitiu	3	3	100.0%
relatiu	1	1	100.0%
S	91	85	93.4%
s.a	4	3	75.0%
sa	1	1	100.0%
sadv	7	7	100.0%
sentence	40	35	87.5%
sn	220	216	98.2%
sp	207	204	98.6%
spec	8	1	12.5%

Tabla 4.15: Precisión de las reglas de detección de núcleos

Al analizar los errores más comunes, se descubrió una discrepancia importante en el etiquetado de las coordinaciones. Durante el desarrollo del trans-

formador se tomó como criterio que el núcleo de la frase en esos casos sería el último de los elementos coordinados, pero en el momento de la evaluación se tomó como criterio que el núcleo de la frase fuera la conjunción. No existen razones muy fuertes para considerar una u otra opción como la mejor. Si tomamos como núcleo el último de los elementos coordinados, estamos tomando un elemento de la clase coordinada, lo cual es útil para tener los rasgos de la clase en el núcleo. Por otro lado, si tomamos como núcleo la conjunción podemos construir un conjunto de rasgos nuevo basado en los elementos coordinados pero con libertad para modificar otros rasgos. Debido a esto, se decidió volver a realizar el cálculo ignorando la influencia de los constituyentes que son conjunciones, lo cual mejoró los resultados, obteniendo una precisión de 98.7%.

La tabla 4.16 muestra la precisión desglosada según el tipo de constituyente luego de eliminar los casos de coordinaciones. Como puede apreciarse, la performance es muy buena para la mayoría de los constituyentes, excepto el caso de los especificadores compuestos. Sin embargo, en general son muy pocos los ejemplos de este tipo de especificadores por lo que la precisión general es muy buena.

Categoría	Total	Correctos	Precisión
grup.a	6	6	100.0%
grup.adv	3	3	100.0%
grup.nom	157	154	98.1%
grup.verb	23	23	100.0%
infinitiu	3	3	100.0%
relatiu	1	1	100.0%
S	85	63	100.0%
s.a	3	3	100.0%
sa	1	1	100.0%
sadv	7	7	100.0%
sentence	35	35	100.0%
sn	216	216	100.0%
sp	204	204	100.0%
spec	8	1	12.5%

Tabla 4.16: Precisión de las reglas de detección de núcleos sin coordinaciones

4.5.2 Clasificación de argumentos

En total se identifica correctamente la relación de un elemento respecto a su núcleo en un 92.5% de los casos. La tabla 4.17 muestra un resumen de la precisión desglosado respecto al tipo de relación.

Como puede apreciarse, la relación más difícil de identificar es la de los complementos, seguida de la de los modificadores. La tabla 4.18 muestra la matriz de confusión para los diferentes tipos de relaciones, donde puede apreciarse que la distinción más difícil para el sistema de reglas es decidir entre clasificar un elemento como complemento o modificador.

Relación	Precisión
Especificador	97.89%
Complemento	84.95%
Modificador	92.86%
Clítico	100.0%
Puntuación	100.0%

Tabla 4.17: Precisión de las reglas de clasificación de argumentos

	Especificador	Complemento	Modificador	Clítico	Puntuación
Especificador	279	3	3	0	0
Complemento	6	333	53	0	0
Modificador	1	18	247	0	0
Clítico	0	0	0	19	0
Puntuación	0	0	0	0	155

Tabla 4.18: Matriz de confusión para la clasificación de argumentos

Además podemos analizar los ejemplos según el tipo de constituyente. En la tabla 4.19 se muestran los resultados agrupados según tipos de constituyentes (por ejemplo los constituyentes `grup.nom` y `sn` se agrupan en `nominal`, los totales sin agrupar se pueden ver en el anexo B). Se puede apreciar que el tipo de constituyente que más influye negativamente en los resultados es el nominal. Si bien su precisión no es la más baja (83.5%) tiene una gran influencia en el resultado debido a que contiene una gran cantidad de ejemplos.

Tipo de constituyente	Total de ejemplos	Ejemplos correctos	Precisión
nominal	267	223	83.5%
verbal	21	21	100.0%
adjetival	11	7	63.6%
adverbial	15	12	80.0%
preposicional	228	227	99.6%
oraciones	365	333	91.2%
otros	9	9	100.0

Tabla 4.19: Clasificación de argumentos según tipo de constituyente

Sabiendo que el tipo de relación más difícil de clasificar es el de los complementos, construimos la tabla 4.20 que muestra la precisión para complementos desglosada por tipo de constituyente. De esta tabla se desprende que los elementos que más influyen en la caída de precisión son los complementos que están dentro de un constituyente nominal (39.7% de precisión sobre 58 ejemplos). Este resultado es esperable ya que en AnCora solamente los nombres deverbales están marcados con complementos, mientras que otros con complementos no están etiquetados.

Tipo de constituyente	Total de ejemplos	Ejemplos correctos	Precisión
nominal	58	23	39.7%
verbal	21	21	100.0%
adjetival	3	0	0.0%
adverbial	3	0	0.0%
preposicional	203	203	100.0%
oraciones	104	86	82.7%

Tabla 4.20: Clasificación de complementos según tipo de constituyente

Capítulo 5

Extracción del léxico

En este capítulo se describe el proceso que se utilizó para extraer las entradas léxicas del corpus transformado y agruparlas en unidades con las que se pudiera trabajar. Las entradas léxicas se agrupan según su comportamiento sintáctico-semántico en unidades que denominamos frames léxicos. Se describe la forma de construcción de los frames léxicos para las tres categorías léxicas más complejas: verbos, nombres y adjetivos.

5.1 Entradas léxicas

Una vez convertido el corpus, el resultado es un nuevo corpus con las siguientes características:

- Todas las reglas son binarias (o unarias).
- Cada constituyente define cuál de sus hijos es el núcleo.
- Se distingue entre dos núcleos: sintáctico y semántico. El núcleo sintáctico guía la concordancia de la expresión con el resto de la oración, mientras que el núcleo semántico guía la estructura argumental de las frases.

Recordemos la definición de nuestra entrada léxica de la sección 3.2.1, donde se indica que cada entrada léxica contiene información sintáctica y semántica acerca de la palabra. Esta información puede resumirse como:

- Categoría gramatical
- Atributos morfológicos para la concordancia
- Valencia combinatoria (sintáctica)
- Estructura argumental (semántica)

Si tenemos en cuenta todas las combinaciones de rasgos que aparecen para cada palabra en el corpus AnCora, la cantidad de entradas léxicas que podemos encontrar en el corpus es muy grande. También es muy dispersa, debido a que cada entrada léxica aparece pocas veces en el corpus. Consideremos los siguientes ejemplos del verbo "comer" que aparecen en el corpus junto con el contexto en el que se usan:

(a) `v5268`: [[Bennett] [se] [**comió**] [a Turner y a quien se puso por delante]]

(b) `v15200`: [[El técnico barcelonista y el mandatario catalán] [**comieron**] [ayer] [juntos] [, con_el_propósito_de aunar fuerzas] [.]]

(c) `v52863`: [[ ] [**Comimos**] [un arroz con pollo muy sabroso] [.]]

(d) `v56148`: [[:] [un solo visón] [se][1] [**comió**] [87 huevos]]

Los ejemplos (a), (c) y (d) son ejemplos de uso del verbo "comer" como verbo transitivo, debido a que tienen un sujeto y un objeto directo, mientras que (b) es un ejemplo de "comer" intransitivo. A su vez, (a) se distingue de los otros dos en el tipo de complemento que aceptan como complemento directo: en un caso es preposicional y en los otros es nominal. Notar que (a) y (d) utilizan la misma forma léxica, con los mismos rasgos morfológicos, pero aun así deben tener entradas léxicas diferentes debido a que aplican una combinatoria diferente.

Las entradas léxicas para los cuatro ejemplos se muestran en las figuras 5.1, 5.2, 5.3 y 5.4.

Analizando la gran cantidad de entradas léxicas diferentes, se propone hacer una primera aproximación al agrupamiento de entradas léxicas en categorías que puedan resultar útiles para el procesamiento.

5.2 Frame léxico

En primer lugar podemos agrupar las palabras del corpus en clases según su lema y categoría gramatical. Esto implica olvidar por el momento el rasgo que define los atributos morfológicos. A los elementos de esta agrupación de entradas léxicas les denominamos *frames léxicos*.

Un *frame léxico* es una estructura de rasgos subespecificada que contiene información sobre: categoría gramatical, valencia sintáctica y estructura argumental. Cada palabra del corpus puede asignarse a un *frame léxico*. Por ejemplo, la figura 5.5 muestra un posible *frame léxico* que corresponde tanto a los ejemplos (c) y (d) de la sección anterior.

El *frame léxico* contiene información acerca de la combinatoria de la palabra, pero a la vez es bastante genérico en cuanto a que esconde los detalles morfológicos. El mismo *frame léxico* de la figura 5.5 puede servir para otras

[1]Este morfema pronominal es absorbido al igual que los clíticos, por lo que su información no es directamente explotable en el momento de la extracción de léxico. Por este motivo, los verbos pronominales se analizarán de la misma manera que los otros verbos.

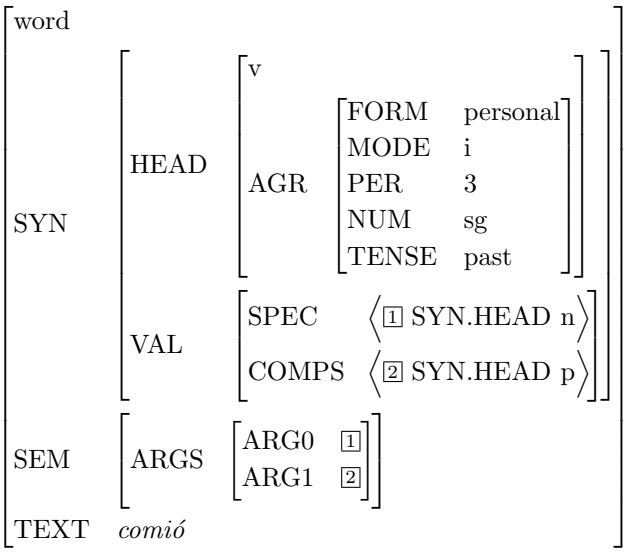

Figura 5.1: Entrada léxica para `v5268`

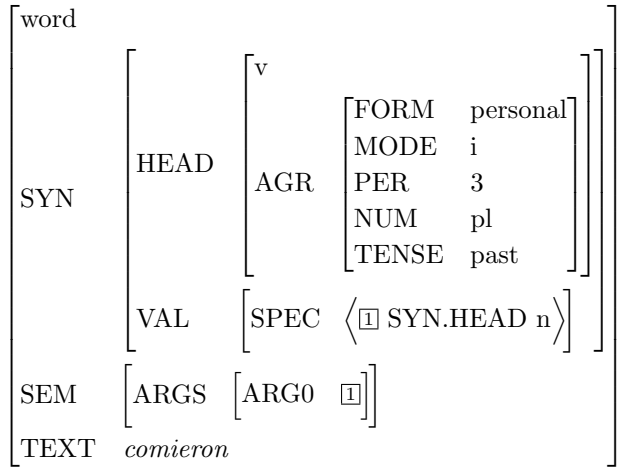

Figura 5.2: Entrada léxica para `v15200`

instancias de verbos transitivos que tengan complemento nominal como "leer" o "mirar".

Dos entradas léxicas que comparten un mismo *frame léxico* tendrán entonces una combinatoria muy similar, compartiendo la categoría gramatical y los tipos de argumentos sintácticos y semánticos. Los *frames léxicos* son una forma de clasificar las palabras del corpus que es más granular que las categorías grama-

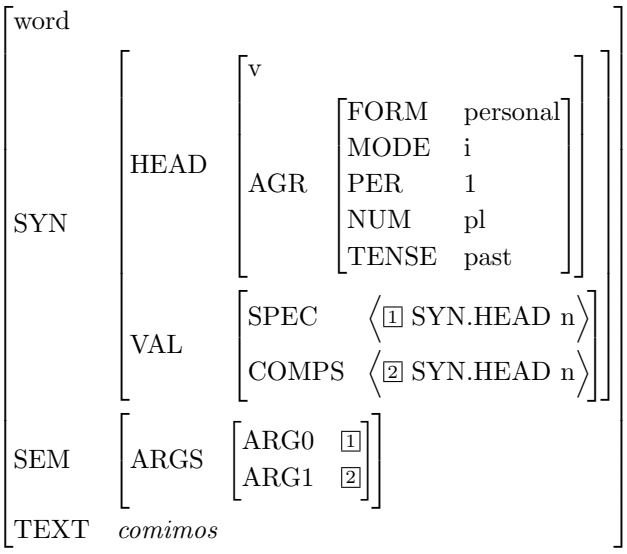

Figura 5.3: Entrada léxica para `v52863`

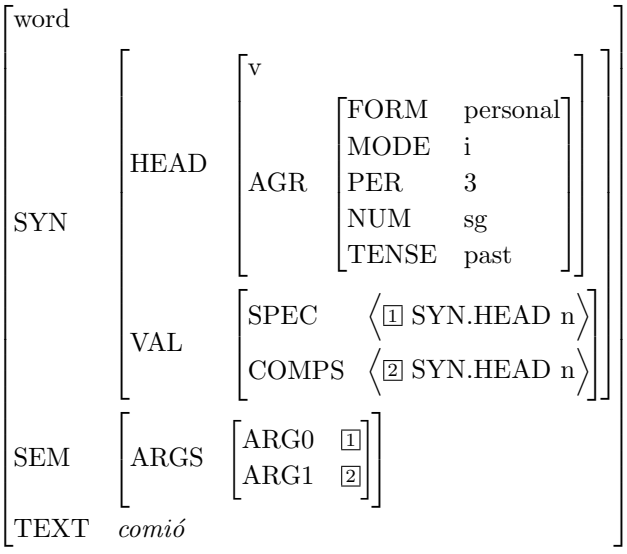

Figura 5.4: Entrada léxica para `v56148`

ticales pero no tan dispersa como las propias entradas léxicas.

El concepto de *frame léxico* que utilizamos es similar al concepto de subcategorización verbal: un agrupamiento de los verbos en clases de equivalencia

$$\begin{bmatrix} \text{word} & & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \text{v} \\ \text{VAL} & \begin{bmatrix} \text{SPEC} & \langle \boxed{1}\ \text{SYN.HEAD n} \rangle \\ \text{COMPS} & \langle \boxed{2}\ \text{SYN.HEAD n} \rangle \end{bmatrix} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{ARGS} & \begin{bmatrix} \text{ARG0} & \boxed{1} \\ \text{ARG1} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figura 5.5: Uno de los *frames léxicos* para el verbo "comer"

en base a qué tipos de complementos aceptan y cuáles son sus características sintáctico-semánticas. En este trabajo utilizamos el término *frame léxico* para incluir las subcategorizaciones tanto verbales como de otras categorías gramaticales. Las subcategorías verbales pueden definirse de manera manual, por ejemplo el proyecto SenSem consta de un corpus con ejemplos de los 250 verbos más frecuentes del español con sus argumentos y adjuntos anotados manualmente [51] y de un léxico que define los *frames* de subcategorización para cada uno de los verbos construido a partir de los ejemplos del corpus [52]. Debido a que la construcción manual de estos *frames* de subcategorización es un trabajo costoso, también se han utilizado técnicas automáticas para realizarla. Por ejemplo en [53] se parte de corpus de gran volumen (3 millones y 50 millones de palabras) inicialmente no etiquetados, se utilizan estrategias estadísticas para identificar los argumentos verbales y a partir de esta información se construyen los *frames* de subcategorización.

Una diferencia importante entre la subcategorización verbal para el inglés y para el español que se menciona habitualmente es que el inglés suele ser más rígido en cuanto al orden de los argumentos verbales, mientras que el español permite cambiar dicho orden más libremente. Por lo tanto para el español se suele optar por utilizar representaciones de subcategorización que ignoren el orden de los argumentos. De esta manera las oraciones "Juan dio caramelos a los niños" y "Juan dio a los niños caramelos" tendrían la misma subcategoría para el verbo. En el presente trabajo también se utiliza una representación independiente del orden de los argumentos en el texto.

5.3 Verbos

La categoría gramatical más compleja en español es la de los verbos, ya que es la que presenta más variedad combinatoria y un paradigma flexivo más rico [54]. Esto significa que para muchos de los verbos puede existir un gran número de *frames léxicos* posibles. Por ejemplo "ir" y "hacer" tienen decenas de *frames léxicos*.

De AnCora se extrajeron 58597 instancias de verbos, de los cuales 54816

tienen algún argumento identificado. Esto significa que para estos verbos el proceso de transformación logró identificar al menos un complemento o el especificador del verbo, sin interesar los modificadores (adjuntos) que pudiera tener.

En total hay 89734 elementos identificados por las heurísticas como argumento de algún verbo, ya sea como complemento o como especificador. Recordemos de la sección 4.2 que los atributos que se utilizan en AnCora para marcar información acerca de los argumentos verbales son `arg` (estructura argumental) y `func` (función gramatical). Sin embargo, estos atributos no siempre están presentes en los argumentos identificados por el proceso. De los 89734 elementos identificados como argumento, 73883 definen el atributo `func` y 67673 definen el atributo `arg`.

Para construir el *frame léxico*, el atributo que más nos interesa es el `arg`, ya que sabiendo el valor de este atributo podemos reconstruir los rasgos `SEM` del *frame*. Solo un 75% de los argumentos tienen identificado el atributo `arg` en AnCora.

Analizando más en detalle los elementos que no tienen definido el atributo `arg`, se pudo detectar una regularidad interesante: en los casos de verbos auxiliares y perífrasis verbales este atributo nunca está definido. Recordemos de la sección 4.4.7 que en AnCora los verbos auxiliares y las perífrasis verbales se encuentran etiquetadas dentro de un constituyente `grup.verb`. Más allá de agruparlos dentro del constituyente, no hay ningún análisis extra. Sin embargo, para poder modelarlos dentro del contexto HPSG es necesario definir cuál de ellos es el dominante y establecer una relación de núcleo-complemento.

En este trabajo consideramos que el núcleo sintáctico es el verbo conjugado, y es necesario marcar de alguna manera al elemento que está actuando como complemento para poder distinguirlo de otros tipos de complementos. Consideramos que los complementos en esta relación no están actuando como ninguno de los argumentos semánticos definidos por la teoría (agente, tema, etc.), por lo que tiene sentido que en AnCora el atributo `arg` no se encuentre etiquetado. Pero debido a que nos interesa poder identificar correctamente estos argumentos, utilizamos una nueva etiqueta `argC` para distinguirlos. De esta manera, el argumento de un verbo auxiliar o una perífrasis va a aparecer en el rasgo semántico `argC` de su núcleo. Por ejemplo, el grupo verbal `grup.verb24` "ha sabido" es una conjugación con verbo auxiliar que se modela como se muestra en la figura 5.6. Notar que:

- El complemento de "ha" es el nodo "sabido", y a su vez es el argumento semántico `argC`.
- El especificador de "ha" aún no está saturado en esta estructura, pero sí está coindizado con el especificador de "sabido" y con el de la frase resultante.
- El complemento que le falta a "sabido" (en este caso una oración subordinada), que además es el tema (`arg1`), pasa a ser el complemento que le falta a la frase resultante.

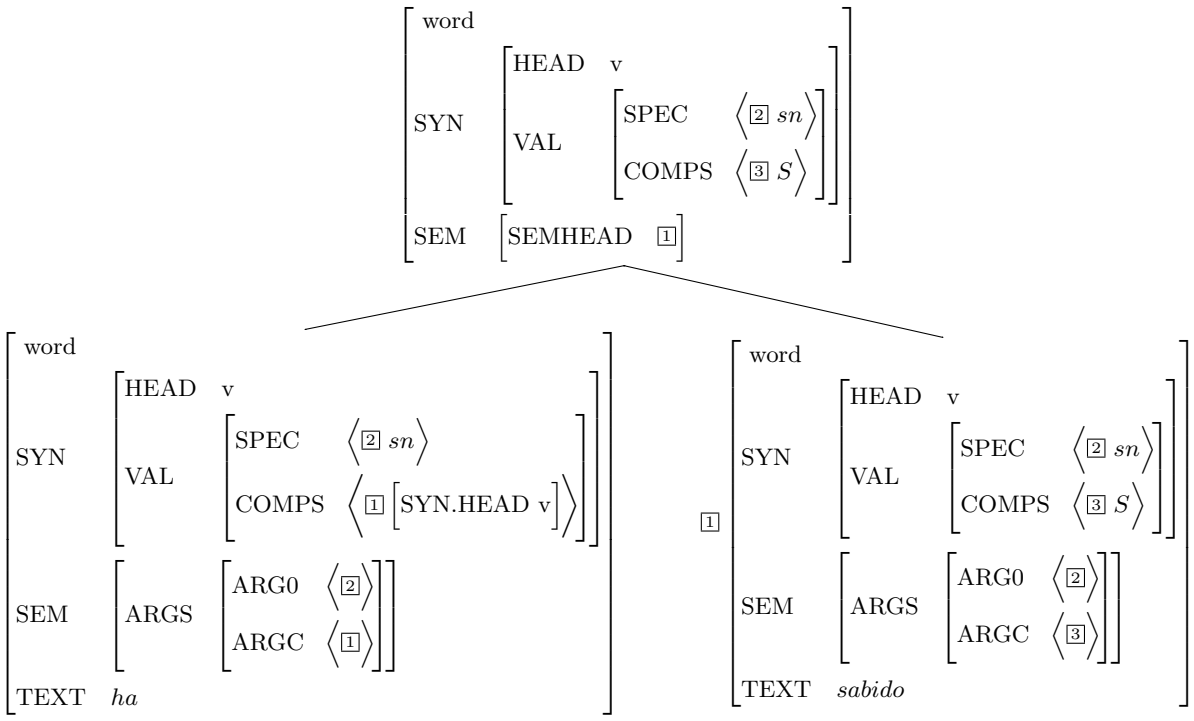

Figura 5.6: Análisis de la perífrasis "ha sabido"

Existen 6210 instancias de verbos auxiliares o perífrasis en AnCora. Al agregar esta nueva marca semántica el total de elementos marcados con el atributo `arg` crece a 82%. Si bien tener esta información más completa es una mejora, aún hay muchos ejemplos de verbos a los que no se les pudo identificar argumentos de ninguna manera.

5.4 Nombres

El *frame léxico* más básico que puede tener un nombre es el que se muestra en la figura 5.7: sin complementos y con un determinante como especificador (por ejemplo "perro"). Este *frame léxico* es compartido por la mayoría de los nombres del corpus. Sin embargo, los nombres en español pueden tener complementos, por ejemplo "la guerra de Vietnam" o "la compra de la empresa por parte de los accionistas". AnCora contiene información acerca de los complementos de nombres, pero solo en el caso de que sean nombres deverbales. Los nombres deverbales son nombres que derivan de verbos y la estructura de sus complementos es muy parecida a la estructura argumental original del verbo con algunas variantes. Por ejemplo: los complementos de nombre siempre son preposicionales y no existe la noción de sujeto, el sujeto verbal se transforma en un complemento preposicional más.

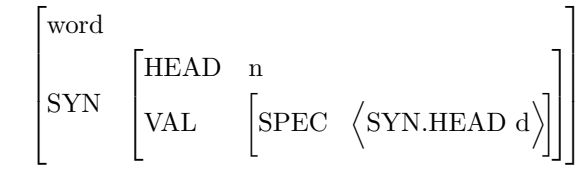

Figura 5.7: *Frame léxico* del nombre común sin complementos

En AnCora hay más de 120.000 instancia de nombres, de los cuales solamente 8270 tienen algún complemento. Todos los nombres con complementos marcados son deverbales, por ejemplo: "preocupación", "funcionamiento" y "adopción". La proporción de nombres con complementos etiquetados es bastante baja, sin embargo casi todos los complementos de nombre etiquetados en el corpus tienen definido el atributo `arg` de estructura argumental, a diferencia de lo que ocurría en el caso de los verbos: hay 8894 elementos identificados como complementos de nombre, de los cuales 8845 tienen el atributo `arg` marcado.

Por ejemplo, el nombre `n16` "construcción", que ocurre en la frase "[[$_{S}$la] [[$_{H}$construcción] [$_{C}$de Altamira_2] [$_{M}$, al norte de Tampico]]], tiene el *frame léxico* que se muestra en la 5.8.

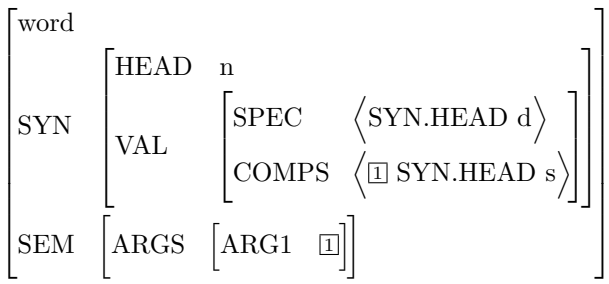

Figura 5.8: *Frame léxico* para el nombre "construcción"

5.5 Adjetivos

El *frame léxico* del adjetivo básico se muestra en la 5.9: un adjetivo no tiene complementos ni especificador, y actúa como modificador de una estructura de tipo nominal. Por ejemplo el adjetivo "rojo".

$$\begin{bmatrix} \text{word} & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \text{a} \\ \text{VAL} & \begin{bmatrix} \text{MOD} & \left\langle \text{SYN.HEAD n} \right\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figura 5.9: *Frame léxico* del adjetivo común sin complementos

Además de estos adjetivos básicos, en español existen adjetivos con argumentos, por ejemplo "difícil de entender", pero en AnCora solamente se marcan los argumentos de un adjetivo si el adjetivo viene de un participio (por ejemplo "elegida" o "encontrados"). Estos adjetivos heredan la estructura argumental del verbo del que partieron.

En total hay 35920 ejemplos de adjetivos en AnCora. 1998 de estos adjetivos tienen marcados los argumentos en el corpus, y de estos 1941 definen el atributo `arg` que se utiliza para determinar la estructura argumental.

Por ejemplo, el adjetivo `a187` "recogido", que ocurre en la frase [[$_{\text{H}}$recogida] [$_{\text{M}}$los últimos días] [$_{\text{C}}$por el buque científico " Akademik_Mstislav_Keldish "] [$_{\text{C}}$en el lugar del accidente]], tiene el *frame léxico* que se muestra en la figura 5.10.

$$\begin{bmatrix} \text{word} & \\ \text{SYN} & \begin{bmatrix} \text{HEAD} & \text{a} \\ \text{VAL} & \begin{bmatrix} \text{MOD} & \left\langle \text{SYN.HEAD n} \right\rangle \\ \text{COMPS} & \left\langle \boxed{1}\ \text{SYN.HEAD s}, \boxed{2}\ \text{SYN.HEAD s} \right\rangle \end{bmatrix} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{ARGS} & \begin{bmatrix} \text{ARG0} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figura 5.10: *Frame léxico* para el adjetivo "recogido"

5.6 Léxico extraído

La tabla 5.1 muestra un resumen de la cantidad de instancias en el corpus para cada categoría. Se muestra el número total de instancias y la cantidad de lemas distintos por categoría. Para las tres categorías agrupadas en *frames léxicos* se muestra además la cantidad de *frames* diferentes extraídos.

Categoría	Instancias	Lemas	Frames
adjetivo (`a`)	34247	4653	49
conjunción (`c`)	25617	116	
determinante (`d`)	72846	90	
puntuación (`f`)	62069	20	
interjección (`i`)	76	32	
nombre (`n`)	116086	18818	161
pronombre (`p`)	21458	82	
adverbio (`r`)	17750	980	
preposición (`s`)	76578	340	
verbo (`v`)	58597	2375	725
fecha (`w`)	2621	656	
número (`z`)	5217	1741	

Tabla 5.1: Cantidad de ejemplos de cada categoría en el léxico

Capítulo 6

Supertagging

Este capítulo describe la construcción del supertagger que se utiliza para etiquetar automáticamente cada palabra de una oración con su frame léxico correspondiente. Primero se discute la necesidad de construir un supertagger y se muestran ejemplos del uso de estas herramientas para diferentes parsers estadísticos. Luego se muestran los diferentes experimentos de supertaggers que se construyeron en el presente trabajo a partir del corpus transformado y la evaluación de performance de los mismos. Finalmente, se describe el supertagger unificado construido para etiquetar los frames léxicos de las categorías gramaticales consideradas.

6.1 Motivación

Para obtener el árbol de *parsing* de una oración, un *parser* necesitará primero definir cuáles son los *frames léxicos* de las palabras. Debido a la gran cantidad de *frames léxicos* posibles para cada palabra, se decidió desarrollar un *supertagger* que permita identificar cuáles de ellos son más probables dada una oración.

Un *supertagger* es similar a un *tagger* en el sentido de que clasifica todas las palabras de una oración en una serie de categorías (a las categorías se les denomina *tags* o etiquetas). La diferencia es que las etiquetas de un *supertagger* incorporan mucha más información que las de un etiquetador sintáctico simple.

En nuestro caso, nos interesa entrenar un *supertagger* que asigne el *frame léxico* correcto para cada palabra. Para lograr esto, es necesario que la etiqueta tenga codificada, además de la categoría gramatical, toda la información sobre la valencia sintáctica y la estructura argumental del *frame*. La forma en que se mapea los *frames léxicos* a etiquetas del *supertagger* será similar para cada categoría gramatical pero con ciertas diferencias. Por ejemplo:

- ofrecer — `v-arg0s_sn-arg1_sn-arg2_sp_a`
  Esta etiqueta del verbo "ofrecer" indica que tendrá tres argumentos: el argumento `arg0` (agente) corresponderá con el sujeto y será un sintagma

nominal; el argumento `arg1` (tema) será un sintagma nominal; y el argumento `arg2` (benefactor) será un sintagma preposicional que utiliza la preposición "a".

- compra — `n-arg1_sp_de`
  Esta etiqueta del nombre "compra" indica que tendrá un argumento `arg1` (tema), que será un sintagma preposicional que utiliza la preposición "de".

- creado — `a-arg0_sp_por`
  Esta etiqueta del adjetivo "creado" (viene del participio del verbo crear) indica que el argumento `arg0` (agente) es un sintagma preposicional que utiliza la preposición "por".

6.2 Antecedentes de supertagging

Como se mencionó en la sección 2.2.3, el *supertagging* es una etapa importante en el *parsing* de gramáticas lexicalizadas complejas (como lo son HPSG, TAG y CCG) debido a que restringe el espacio de búsqueda que el *parser* necesitará recorrer para obtener el mejor análisis. Como desventaja, si el desempeño del *supertagger* no es bueno, es posible que se descarte la solución óptima y por lo tanto el *parser* no pueda encontrar el mejor análisis.

El concepto de *supertagging* se describe por primera vez en [35], donde se utiliza como paso previo a la ejecución de un *parser* LTAG. Para la construcción de *supertaggers* pueden utilizarse los mismos algoritmos que se utilizan para el *tagging* morfosintáctico. Por ejemplo, el *supertagger* para LTAG mencionado utiliza un modelo de trigramas y su *accuracy*[1] es de 68%. Los autores además indican que podrían mejorar la performance y llevarla a 77% si incorporan información sobre dependencias de largo alcance.

Se ha aplicado la técnica de *supertagging* en el contexto de otros tipos de *parsers* lexicalizados. En [55] se describe la utilización de un *supertagger* para un *parser* CCG para el idioma inglés. Lo novedoso de la técnica es que permiten que el *supertagger* devuelva, además de la etiqueta más probable, otras etiquetas candidatas con menor probabilidad. De esta manera, si el *parser* no encuentra un buen análisis puede intentar con otras entradas léxicas candidatas. El *supertagger* devuelve la etiqueta más probable y todas las que estén como máximo dentro de un valor β de distancia. Reportan que el *supertagger* construido tiene un 97,7% de *accuracy* y en promedio devuelve 1,4 etiquetas por palabra.

Para el caso de HPSG para el idioma inglés, destacamos el uso de *supertaggers* en el contexto del *parser* PET y el *parser* Enju. En [56] se describe la utilización de un *supertagger* para mejorar la performance del *parser* PET. El *supertagger* construido tiene 90,2% de *accuracy* para la mejor etiqueta, si se consideran las dos mejores etiquetas la performance sube al 97,2%. El *supertagger* de Enju [34] está basado en la técnica del perceptrón, pero incorporan como

[1] *Accuracy* o exactitud: cantidad de ejemplos etiquetados correctamente sobre total de ejemplos etiquetados.

feature si es posible parsear la oración utilizando una gramática CFG construida a partir de su gramática HPSG siguiendo la técnica descrita en [57]. El lenguaje reconocido por la CFG es un superconjunto del reconocido por la HPSG, pero debido a su simplicidad es mucho más rápido de parsear. El resultado es que la CFG permite descartar tempranamente muchos análisis inválidos (falsos positivos), lo cual lleva a un 93,98% de *accuracy*.

En [58] se describen experimentos de *supertagging* para la gramática HPSG LXGram para el portugués. Entrenan *supertaggers* utilizando tres modelos (Hidden Markov Model, Support Vector Machine y Log Linear). El mejor *supertagger* entrenado obtiene 92,63% de *accuracy* sobre verbos y 95,18% sobre nombres, entrenado sobre un corpus de 21.000 oraciones.

Para el español, el único ejemplo de *supertagging* que conocemos es el descrito en [59], un *supertagger* para una gramática LTAG para el español. En el mencionado trabajo transforman el corpus AnCora a un conjunto de árboles LTAG y utilizan esta información para entrenar un *supertagger* de tipo MaxEnt. Reportan un 79,64% de *accuracy*.

6.3 Verbos

Observemos el ejemplo del verbo "ofrecer", considerando la etiqueta que definimos anteriormente:

- ofrecer — `v-arg0s_sn-arg1_sn-arg2_sp_a`

A partir de esta etiqueta podemos deducir que esta instancia del verbo "ofrecer" tendrá tres argumentos: uno de ellos es el especificador, es un sintagma nominal y será el agente (`arg0`), un complemento nominal que será el tema (`arg1`) y un complemento preposicional (que tiene como núcleo la preposición "a") que será el benefactor (`arg2`). La etiqueta no considera información posicional de los constituyentes, o sea que se aplicará tanto al verbo de la oración "Juan ofreció una manzana a los niños" como al de la oración "a los niños Juan les ofreció una manzana". Notar que la información del clítico "les" no es tenida en cuenta en este análisis, se decidió dejar fuera del alcance de este trabajo la complejidad de la identificación de los argumentos saturados por clíticos.

Podemos utilizar la información de esta etiqueta para reconstruir el *frame léxico* asociado a esta instancia del verbo "ofrecer". El *frame* se muestra en la figura 6.1.

A su vez, incluyendo la información morfológica de la palabra "ofreció" podemos reconstruir la entrada léxica correcta para el verbo en "Juan ofreció una manzana a los niños".

6.3.1 Corpus

Las 58597 instancias de verbos se agruparon en 725 *frames léxicos* diferentes según sus rasgos de especificador, complementos y estructura argumental. A

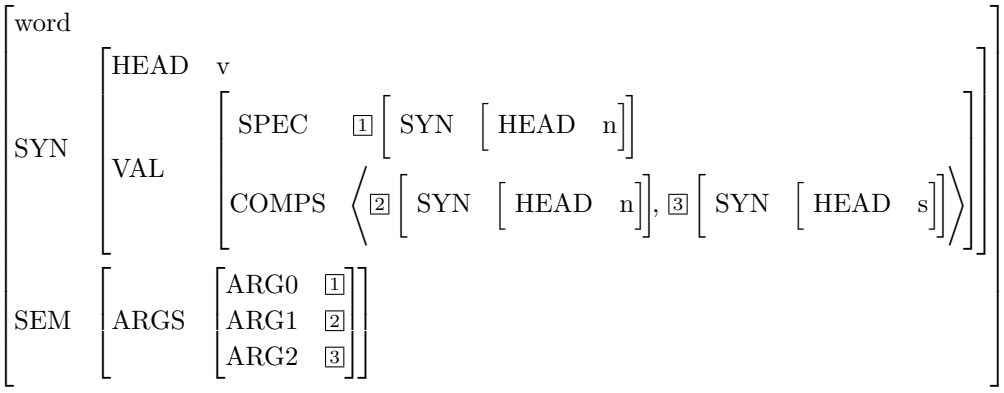

Figura 6.1: *Frame léxico* para el verbo "ofrecer"

partir de esto se construyó un corpus de oraciones etiquetadas que incluye las palabras y una etiqueta para cada palabra. La etiqueta puede ser:

- Si la palabra es un verbo, se utiliza la etiqueta del *frame léxico*.
- En caso contrario, se utiliza la categoría gramatical provista por AnCora.

El corpus contiene 17375 oraciones (516843 palabras), y se dividió en tres conjuntos:

- Conjunto de entrenamiento: 15610 oraciones (471389 palabras).
- Conjunto de desarrollo: 772 oraciones (22926 palabras).
- Conjunto de test: 993 oraciones (22528 palabras).

Existe una gran cantidad de etiquetas (*frames léxicos*), algunas de ellas aparecen muchas veces en el corpus, mientras que otras aparecen muy pocas. Es difícil que un algoritmo de aprendizaje automático logre extraer información correcta de las etiquetas que aparecen muy pocas veces. Debido a eso, se sustituyó todas las etiquetas que aparecieran menos de 30 veces en el corpus por una etiqueta genérica `v-unknown`. De esta manera la cantidad de etiquetas verbales diferentes se redujo a 92.

Se calculó una *baseline* para la performance del etiquetado de este corpus de la siguiente manera: En el corpus de entrenamiento se calcula la frecuencia de todas las etiquetas para cada uno de los lemas. El etiquetador asigna las n etiquetas más frecuentes para los lemas del corpus de desarrollo. La tabla 6.1 muestra la exactitud (*accuracy*) de este etiquetador para diferentes valores de n.

Se realizó una serie de experimentos [60] utilizando clasificadores de Maximum Entropy [61] y Conditional Random Fields [62] sobre el corpus generado. Se entrenaron clasificadores utilizando diferentes variantes del corpus y diferentes conjuntos de *features* para el aprendizaje, como se explica a continuación.

# etiquetas más frecuentes	1	2	3	4	5
todas las categorías	88.83	93.59	95.25	96.47	97.16
solo verbos	24.76	44.12	58.59	69.52	75.83

Tabla 6.1: *Accuracy* de la línea base para verbos

6.3.2 Lemas

El primer conjunto de experimentos utiliza solamente los lemas de cada una de las palabras del corpus como *features* para el aprendizaje. Se entrenaron diferentes clasificadores considerando las siguientes variables:

- contexto: Denominamos `contexto` al tamaño de la ventana de palabras alrededor de la palabra que se quiere etiquetar. Un valor i para esta variable representa que para etiquetar una palabra se utiliza la información de las $(i-1)/2$ palabras anteriores, la palabra a etiquetar, y las $(i-1)/2$ palabras posteriores. Los valores posibles para esta variable en los experimentos realizados fueron 5, 7, 9 y 11.

- umbral: Esta variable es el `umbral` de palabras infrecuentes. Se construyeron diferentes versiones del corpus donde todas las palabras que aparezcan menos de `umbral` veces son sustituidas por un *token* de palabra desconocida. Este *token* depende de la categoría gramatical de la palabra. Por ejemplo los sustantivos infrecuentes se marcan como `unknown_n` y los verbos infrecuentes como `unknown_v`. Los valores posibles para esta variable en los experimentos realizados fueron 20, 30 y 50. Se realizaron experimentos preliminares utilizando valores de `umbral` de 10 y 15, pero su performance resultó menor que en los otros casos, por lo que para los experimentos finales se decidió concentrarse en los otros valores.

Los experimentos con este corpus se hicieron solamente entrenando clasificadores tipo CRF. El principal problema encontrado es que el entrenamiento de estos clasificadores puede ser muy lento (por ejemplo el modelo que tomó más tiempo tardó 43 horas en entrenar). Para paliar este problema, se crearon versiones del corpus más pequeñas (de 50.000 y 106.000 *tokens*) y se entrenaron primero los clasificadores en las versiones más pequeñas. Luego se pasó a entrenar los clasificadores más grandes solamente para las configuraciones que parecían más prometedoras en el sentido de que dieron mejores resultados sobre los corpus reducidos.

Del conjunto de etiquetadores entrenado, el que obtuvo mejor performance fue el que utiliza `contexto` 5 y `umbral` 30. La exactitud de este etiquetador sobre el corpus de desarrollo puede verse en la tabla 6.2. La performance detallada de los clasificadores entrenados para esta etapa sobre los corpus de distintos tamaños se puede encontrar en el anexo C.1.1.

# etiquetas más frecuentes	1	2	3	4	5
todas las categorías	93.15	96.37	97.67	98.43	98.81
solo verbos	48.79	67.57	78.05	83.58	86.75

Tabla 6.2: *Accuracy* para el clasificador CRF usando lemas como *features*

6.3.3 Lemas y categorías gramaticales

El segundo conjunto de experimentos utiliza los lemas y la categoría gramatical (*POS tag* precalculado) de cada una de las palabras del corpus como *features* para el aprendizaje. La idea detrás de estos experimentos es que al tener un valor precalculado para la categoría gramatical, el clasificador no asignará erróneamente las palabras de otras categorías como verbos (y viceversa). Sabiendo de antemano cuáles son los verbos, el clasificador podría concentrarse en elegir dentro de los *frames* verbales.

Al igual que en el caso anterior, se entrenaron diferentes clasificadores para diferentes valores de `contexto`, `umbral`, y tamaño del corpus. En este caso se entrenaron dos juegos de clasificadores, utilizando modelos MaxEnt y CRF.

Los clasificadores CRF siguieron una estrategia similar a la etapa anterior: se entrenaron modelos en corpus más pequeños y se fue seleccionando los más prometedores para entrenarse sobre corpus más grandes. En cambio para MaxEnt, debido a que su tiempo de entrenamiento es mucho menor, se entrenaron clasificadores utilizando el corpus completo de entrenamiento.

La configuración con los mejores resultados para CRF sobre el corpus de desarrollo es utilizando `umbral` 30 y `contexto` 7. La tabla 6.3 muestra la performance de este clasificador sobre el corpus de desarrollo y sobre el corpus de test.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	94.57	96.90	98.11	98.80	99.12
	solo verbos	51.75	71.41	81.51	87.51	90.20
corpus de test	todas las categorías	93.51	96.30	97.74	98.50	98.91
	solo verbos	51.53	71.09	81.35	86.82	89.83

Tabla 6.3: *Accuracy* para el clasificador CRF usando lemas y *POS tags* como *features*

La configuración con los mejores resultados para MaxEnt sobre el corpus de desarrollo es utilizando `umbral` 30 y `contexto` 5. La tabla 6.4 muestra la performance de este clasificador sobre el corpus de desarrollo y sobre el corpus de test.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	94.03	96.32	97.68	98.37	98.77
	solo verbos	45.59	66.49	78.90	85.17	88.79
corpus de test	todas las categorías	92.95	95.61	97.11	97.89	98.38
	solo verbos	49.03	68.22	79.07	84.71	88.28

Tabla 6.4: *Accuracy* para el clasificador CRF usando lemas y *POS tags* como *features*

6.3.4 Solamente verbos con argumentos

Al analizar los resultados de los diferentes *taggers* entrenados, se notó que una gran cantidad de errores se producen cuando el *tagger* asigna la etiqueta `v` a un verbo que debería tener alguna categoría más específica. Los casos de verbos a los cuales se les asigna la etiqueta `v` se pueden dividir en dos clases:

- Se asigna la etiqueta `v` si la etiqueta a asociar al verbo no es muy frecuente en el corpus. Se decidió limitar el mínimo de apariciones de una categoría debido a que es difícil aprender clases poco numerosas, y en estos casos se sustituye la etiqueta por una `v` genérica.
- Se asigna la etiqueta `v` si en AnCora no existe ningún elemento marcado como argumento para ese verbo, por lo tanto al construir la etiqueta la única información conocida es la categoría gramatical `v`.

El primer caso aparece 1124 veces en el corpus, mientras que el segundo aparece 11246 veces. El segundo caso a su vez puede ocurrir por dos motivos:

- El verbo efectivamente no tiene ningún argumento. Esto es válido en español ya que existe una clase de verbos que no llevan sujeto ni complementos, denominados verbos impersonales. Por ejemplo los verbos atmosféricos como "llover" o "nevar" son de este tipo. Sin embargo, no se encontraron ejemplos del uso de este tipo de verbos en el corpus.
- Los argumentos del verbo no están en el corpus o no pudieron recuperarse luego del proceso de transformación. Un caso en el que esto ocurre es el de los verbos que aparecen dentro de una subordinada relativa: `S536` - "que volaron sobre una flota de buques pesqueros que operaba en aguas del Atlántico_Sur", al transformar esta oración, el pronombre "que" es separado de la oración relativa, por lo que se necesitaría más procesamiento para interpretarlo como argumento. Esto quedó fuera del alcance de estos experimentos.

El hecho de que muchos de los argumentos de los verbos no estén debidamente etiquetados es un problema a la hora de aprender de este corpus. Para evitar este problema, se creó una nueva versión del corpus eliminando todos los ejemplos de verbos que no contienen ningún argumento. Esto estaría eliminando

además todos los ejemplos de verbos impersonales, pero como ya se mencionó no se encontraron instancias de este tipo de verbos.

Se decidió eliminar las oraciones enteras que contuvieran algún verbo sin argumentos marcados. Este enfoque es un poco agresivo ya que elimina además muchas instancias de verbos que sí tienen sus argumentos identificados pero que ocurren en la misma oración en que hay otro verbo sin argumentos. Sin embargo, es necesario realizar este podado debido a que de lo contrario se hubiera requerido analizar manualmente cada verbo y su contexto dentro de cada oración para separar los que tuvieran todos sus argumentos de los que no los tuvieran. El corpus resultante tiene aproximadamente la mitad de las oraciones de AnCora. El nuevo corpus contiene 9920 oraciones (258722 palabras), y se dividió en tres conjuntos:

- Conjunto de entrenamiento: 8663 oraciones (233105 palabras).
- Conjunto de desarrollo: 589 oraciones (12471 palabras).
- Conjunto de test: 668 oraciones (13146 palabras).

Se realizaron experimentos creando clasificadores de tipo MaxEnt y CRF. Los valores de `umbral` que se probaron en estos experimentos fueron 20, 30 y 50. Los valores de `contexto` fueron 5, 7, 9 y 11. En el caso de CRF, se crearon versiones más pequeñas del corpus (de 60.000 y 120.000 palabras) para poder entrenar rápidamente modelos con las distintas configuraciones, y luego entrenar los más prometedores sobre el corpus total.

De los clasificadores CRF construidos, el máximo de performance se alcanzó utilizando `contexto` 7 y `umbral` 20. Los resultados de este clasificador pueden verse en la tabla 6.5. Los resultados de todos los experimentos para las diferentes combinaciones de variables pueden encontrarse en el anexo C.1.3.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	95.64	97.53	98.20	98.64	98.85
	solo verbos	57.33	75.84	82.35	86.67	88.78
corpus de test	todas las categorías	95.84	97.51	98.21	98.61	98.94
	solo verbos	61.77	77.15	83.58	87.21	90.29

Tabla 6.5: *Accuracy* para el clasificador CRF entrenado solo para verbos con argumentos

Para el caso de MaxEnt, el mejor clasificador entrenado utiliza `contexto` 5 y `umbral` 30. Sus resultados pueden verse en la tabla 6.6. Como puede apreciarse, los valores de performance para este clasificador sobre el corpus de desarrollo son muy similares a los alcanzados con el mejor clasificador de CRF. Sin embargo, el mismo clasificador sobre el corpus de test termina dando valores algo inferiores.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	95.48	97.24	98.20	98.58	98.81
	solo verbos	55.76	73.02	82.35	86.12	88.31
corpus de test	todas las categorías	95.54	97.21	97.90	98.46	98.78
	solo verbos	59.05	74.35	80.71	85.81	88.75

Tabla 6.6: *Accuracy* para el clasificador MaxEnt entrenado solo para verbos con argumentos

6.4 Nombres

La forma de codificar la estructura de argumentos de los nombres en una etiqueta es similar a la utilizada para el caso de verbos, con la simplificación de que no puede existir un argumento marcado como sujeto. Por ejemplo:

- El grupo nominal `grup.nom211` es "transferencia de acciones a AUNA , (...)", su núcleo nominal "transferencia" queda etiquetado con la categoría `n-arg1_sp_de-arg2_sp_a` porque tiene dos argumentos preposicionales:
  - uno comienza con "de" y tiene valor `arg1` (tema): "de acciones"
  - el otro empieza con "a" y tiene valor `arg2` (beneficiario): "a AUNA , (...)"
- El grupo nominal `grup.nom99068` es "propuesta del PP", su núcleo nominal "propuesta" queda etiquetado con la categoría `n-arg0_sp_de` porque tiene un solo argumento preposicional que comienza con "de" (la preposición "de" es la correspondiente a la contracción "del") y tiene valor `arg0` (agente): "del PP".

Se eligió no incluir la información acerca de los especificadores de nombres para concentrarse solamente en la información de estructura argumental de los mismos. En español los especificadores de nombres suelen ser determinantes y para nombres comunes pueden estar presentes o no, mientras que para nombres propios es común que no se utilicen determinantes (aunque hay excepciones).

6.4.1 Corpus

Los 8270 nombres etiquetados se agrupan en 161 *frames léxicos* según sus rasgos. Debido a que la cantidad de nombres etiquetados con complementos no es tan grande, resulta difícil que los clasificadores aprendan las clases que tienen pocos ejemplos, por lo que se decidió limitar la cantidad de ejemplos por *frame*: si un *frame* aparece menos de 20 veces se lo sustituye por un *frame* genérico `n-unknown`. Esto hace que la cantidad de *frames* diferentes baje a 27.

Se partió el corpus original en tres conjuntos de la misma manera que se hizo para el corpus de verbos: conjunto de entrenamiento (de 47000 palabras aproximadamente), conjuntos de desarrollo y test (de 23000 palabras aproximadamente cada uno).

De manera similar a lo realizado para el caso de los verbos, se tomó las etiquetas más frecuentes en el conjunto de entrenamiento y se las aplicó a los ejemplos del conjunto de desarrollo para calcular una *baseline* de performance sobre este corpus. La misma se muestra en la tabla 6.7.

# etiquetas más frecuentes	1	2	3	4	5
todas las categorías	95.25	99.14	99.39	99.51	99.56
solo nombres	13.42	65.18	76.36	84.03	87.86

Tabla 6.7: *Accuracy* de la línea base para nombres

6.4.2 Experimentos

Para estos experimentos se construyeron clasificadores de tipo CRF y MaxEnt utilizando las siguientes variantes: los valores de `umbral` son 10, 20 y 30; los valores de `contexto` son 5, 7, 9 y 11. Los resultados de dichos experimentos están en el anexo C.2.1.

El clasificador CRF con mejor desempeño sobre el corpus de desarrollo es el que utiliza `umbral` 10 y `contexto` 5, aunque considerando mayor cantidad de candidatos pasa a ser mejor el que utiliza `umbral` 20 y `contexto` 9. Se consideró el primero como el mejor clasificador para ejecutarlo contra los ejemplos del corpus de test, los resultados se muestran en la tabla 6.8.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	98.87	99.60	99.83	99.88	99.90
	solo nombres	36.10	73.48	88.18	91.37	92.65
corpus de test	todas las categorías	99.34	99.74	99.92	99.96	99.97
	solo nombres	30.99	69.01	91.23	94.15	95.91

Tabla 6.8: *Accuracy* para el clasificador CRF de nombres

En el caso de MaxEnt, notamos que si consideramos un solo candidato, todos los clasificadores MaxEnt son superiores a los CRF entrenados con los mismos datos. Sin embargo esta diferencia se nivela al considerar varios candidatos. El mejor clasificador MaxEnt utiliza `umbral` 10 y `contexto` 5, aunque para mayor cantidad de candidatos pasa a ser mejor el que utiliza `umbral` 10 y `contexto` 11. Se consideró el primero como el mejor clasificador para ejecutarlo contra los ejemplos del corpus de test, los resultados se muestran en la tabla 6.9.

Una diferencia importante entre el corpus de *frames* para verbos y el corpus de *frames* para nombres es que en este último una de las etiquetas se utiliza en la gran mayoría de las instancias de nombres: la etiqueta `n`. Esto se debe a que la gran mayoría de los nombres que aparecen no son deverbales, y por lo tanto para AnCora no tienen complementos. Resulta difícil lograr etiquetar correctamente los ejemplos de nombres en categorías cuando una de las posibles categorías se lleva casi todas las instancias: hay 111167 ejemplos de nombres, de los cuales solo 7760 (menos del 7%) pertenecen a alguna categoría con complementos. Los

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	98.80	99.55	99.79	99.87	99.92
	solo nombres	40.26	71.57	85.62	91.05	94.25
corpus de test	todas las categorías	99.31	99.74	99.90	99.95	99.96
	solo nombres	37.43	73.10	87.72	92.98	94.15

Tabla 6.9: *Accuracy* para el clasificador MaxEnt de nombres

resultados de todos los clasificadores entrenados están sesgados a etiquetar como nombres genéricos muchos de los ejemplos que deberían tener una categoría particular.

Dado que no hay ejemplos de un nombre etiquetado como otra categoría que no sea nombre, podemos analizar todas las instancias de nombres distinguiendo entre dos clases distintas: "nombres genéricos" y "subclases de nombre".

Considerando solamente estas dos clases, podemos obtener la precisión y *recall* de la categoría "subclase de nombre". Para el clasificador CRF, la precisión es 0.6739 mientras que el *recall* es 0.3962, lo cual da una medida F1 de 0.2495. Es un resultado muy bajo que se explica principalmente por el bajo *recall*, lo que significa que hay una gran cantidad de nombres que debían etiquetarse como subclase de nombre con complementos, pero quedaron etiquetadas como nombres genéricos. Para el clasificador MaxEnt, la precisión es 0.6337 y el *recall* es 0.4920, lo cual da una medida F1 de 0.2770. Este clasificador obtiene peor precisión, pero el *recall* mejora mucho respecto al de CRF, lo cual alcanza para que la medida F1 sea un poco mejor. El detalle de precisión y *recall* de nombres vs subclases de nombres para cada clasificador puede verse en el anexo C.2.2.

6.5 Adjetivos

Las etiquetas para los *frames léxicos* de adjetivos son similares a las de nombres. Todos los argumentos de los adjetivos que consideramos con preposicionales. En el corpus existen 8 ejemplos en los cuales uno de los argumentos de adjetivo es no preposicional (grupo nominal u oración subordinada). Debido a que es una cantidad muy baja para poder aprender regularidades estadísticas, se decidió no considerar estos casos en los experimentos. Al igual que en el caso de nombres, el sujeto no estará marcado con una etiqueta especial, debido a que será un argumento preposicional más del adjetivo.

Ejemplos:

- La oración subordinada `S1668` es "[[impuesto] [a Irak] [por la ONU] [en 1990]]", su núcleo adjetival "impuesto" queda etiquetado con la categoría `a-arg0_sp_por-arg2_sp_a` porque tiene dos argumentos:
  - uno comienza con "por" y tiene valor `arg0` (agente): "por la ONU"
  - el otro empieza con "a" y tiene valor `arg2` (beneficiario): "a Irak"

- La oración subordinada `S274` es "[[,] [dedicada] [a la " Gobernabilidad y la democracia "] [,]]", su núcleo nominal "dedicada" queda etiquetado con la categoría `a-arg1_sp_a` porque tiene un solo argumento preposicional que comienza con "a" y tiene valor `arg1` (tema): "a la " Gobernabilidad y la democracia "".

6.5.1 Corpus

Los 1998 adjetivos etiquetados se agrupan en 49 *frames léxicos* según sus rasgos. Para evitar tener categorías con muy pocos ejemplos, se suavizó el corpus limitando la cantidad de veces mínimas que puede aparecer un *frame* a 10, lo cual reduce la cantidad de *frames* diferentes a 13.

Se partió el corpus original en tres conjuntos de la misma manera que se hizo para el corpus de verbos: conjunto de entrenamiento (de 47000 palabras aproximadamente), conjuntos de desarrollo y test (de 23000 palabras aproximadamente cada uno). De manera similar a lo realizado para el caso de los verbos y los nombres, se tomó las etiquetas más frecuentes en el conjunto de entrenamiento y se las aplicó a los ejemplos del conjunto de desarrollo para calcular una *baseline* de performance sobre este corpus. La misma se muestra en la 6.10.

# etiquetas más frecuentes	1	2	3	4	5
todas las categorías	96.31	99.46	99.62	99.67	99.68
solo adjetivos	21.25	66.25	76.25	83.75	87.50

Tabla 6.10: *Accuracy* de la línea base para adjetivos

6.5.2 Experimentos

Para estos experimentos se construyeron clasificadores de tipo CRF y MaxEnt utilizando las siguientes variantes: los valores de `umbral` son 10, 20 y 30; los valores de `contexto` son 5, 7, 9 y 11. Los resultados de dichos experimentos están en el anexo C.3.

El clasificador CRF con mejor desempeño sobre el corpus de desarrollo es el que utiliza `umbral` 10 y `contexto` 5. La performance de este clasificador sobre el corpus de desarrollo y de test puede verse en la tabla 6.11.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	99.79	99.95	99.97	99.97	99.98
	solo adjetivos	66.25	86.25	90.00	92.50	95.00
corpus de test	todas las categorías	99.76	99.92	99.96	99.98	99.98
	solo adjetivos	45.61	71.93	84.21	91.23	92.98

Tabla 6.11: *Accuracy* para el clasificador CRF de adjetivos

El clasificador MaxEnt con mejor desempeño sobre el corpus de desarrollo es el que utiliza `umbral` 10 y `contexto` 5. La performance de este clasificador

sobre el corpus de desarrollo y de test puede verse en la table 6.12.

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	99.76	99.94	99.97	99.98	99.98
	solo adjetivos	65.00	85.00	91.25	93.75	95.00
corpus de test	todas las categorías	99.76	99.92	99.95	99.97	99.97
	solo adjetivos	54.39	75.44	80.70	87.72	89.47

Tabla 6.12: *Accuracy* para el clasificador MaxEnt de adjetivos

6.6 Supertagger unificado

Los clasificadores descritos hasta el momento funcionan de manera independiente. Cada uno de ellos se enfoca en clasificar los *frames léxicos* para una sola categoría gramatical (verbos, nombres o adjetivos). De manera de simplificar el proceso de *supertagging*, se construyó un clasificador que permite etiquetar en una sola aplicación los *frames* de las tres categorías gramaticales a la vez.

Los mejores *supertaggers* construidos para las diferentes categorías gramaticales difieren en los valores de `contexto` y `umbral`, y también en la presencia o ausencia de oraciones con verbos sin argumentos etiquetados. Pero para construir un *supertagger* unificado se debe seleccionar solamente una versión del corpus y un único conjunto de *features* de entrenamiento. Debido a que los verbos son la categoría gramatical más compleja desde el punto de vista combinatorio, se decidió tomar como referencia los mejores clasificadores construidos para verbos y utilizar estas configuraciones para construir *supertaggers* unificados, teniendo en cuenta que esto puede implicar que la performance para las otras categorías gramaticales no sea la óptima obtenida anteriormente.

Se entrenaron cuatro nuevos *supertaggers* con las configuraciones que se muestran en la tabla 6.13. Dichas configuraciones están basadas en las que dieron mejores resultados para el etiquetado de *frames* verbales.

Modelo	CRF	MaxEnt	CRF	MaxEnt
Solo verbos con argumentos	no	no	si	si
Umbral	30	30	20	20
Contexto	7	5	7	7

Tabla 6.13: Atributos que se utilizaron para entrenar los *supertaggers* unificados

El *supertagger* que obtuvo un mejor rendimiento en promedio es el que se entrenó con modelo CRF, usando `umbral` 20 y `contexto` 7, sobre la versión del corpus que solamente tiene verbos con algún argumento etiquetado. Sus resultados se muestran en la tabla 6.14. Los resultados para los demás *supertaggers* unificados pueden verse en el anexo C.4.

Como puede apreciarse, el *supertagger* unificado obtiene un desempeño para verbos similar al obtenido en el experimento de etiquetado con categoría gra-

matical única (83,58% considerando las tres etiquetas más probables). Para los nombres, la performance del *supertagger* unificado es menor que el óptimo obtenido (85,78% contra 91,23% para las tres etiquetas más probables). Para los adjetivos, la performance es también un poco menor, pero la diferencia es menos marcada que para los nombres (81,40% contra 84,21% para las tres etiquetas más probables).

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	94.15	96.85	97.85	98.30	98.64
	solo verbos	59.22	75.22	81.73	85.18	88.08
	solo nombres	26.79	67.46	85.65	91.39	93.30
	solo adjetivos	46.38	88.41	92.75	92.75	94.20
corpus de test	todas las categorías	94.48	96.99	97.93	98.53	98.90
	solo verbos	63.80	77.99	83.58	87.63	90.29
	solo nombres	27.94	65.69	85.78	94.12	97.55
	solo adjetivos	41.86	74.42	81.40	90.70	97.67

Tabla 6.14: *Accuracy* para el *supertagger* unificado: modelo CRF, `umbral` 20, `contexto` 7, solo verbos con argumentos

Capítulo 7

Discusión

En este capítulo se presentan los resultados del trabajo detallando los recursos construidos. Se describen además líneas de investigación que quedaron fuera del alcance pero que sería interesante explorar en el futuro.

7.1 Conclusiones

Como resultado del presente trabajo se generaron diferentes recursos y herramientas necesarios para la construcción de un *parser* estadístico HPSG para el español.

7.1.1 Gramática HPSG para el español

Se definieron las estructura de rasgos a utilizar para la representación de las palabras (`word`) y de las frases (`phrase`). Ambas estructuras de rasgos heredan del supertipo de las expresiones (`expr`).

Se definieron las reglas de combinatoria de expresiones para formar frases. La gramática cuenta con 13 reglas incluyendo: reglas del especificador, reglas de los complementos, reglas de los modificadores y reglas de coordinación. Además se incluyen reglas para simplificar la representación de algunos fenómenos del corpus (reglas de puntuación, clíticos y relativas).

7.1.2 Corpus de oraciones HPSG

Se transformaron las oraciones del corpus AnCora a un formato compatible con la gramática HPSG definida. El corpus resultante utiliza las reglas binarias y marca los núcleos sintácticos y semánticos de cada uno de los constituyentes. Se marcan además los constituyentes que actúan como argumentos sintácticos y semánticos de otros constituyentes.

El proceso de transformación combina un enfoque *top-down* de simplificación de constituyentes complejos con uno *bottom-up* de detección de núcleos y clasificación de argumentos para árboles HPSG elementales. Se utiliza un conjunto de reglas construido manualmente para la detección de núcleos y la clasificación de argumentos.

Considerando el corpus original como si fuera anotado con una gramática libre de contexto, la cantidad de reglas para cada tipo de constituyente puede ser muy alta. Por ejemplo, existen 5.800 reglas para oraciones subordinadas y 900 reglas para grupos nominales. Esta complejidad se vio reflejada en el proceso *top-down*, debido a que las etapas de este proceso intentan simplificar las reglas compuestas (por ejemplo las coordinaciones) para dejar un conjunto de árboles simples. También se vio reflejada en el proceso *bottom-up*, debido a que incluso los árboles simples tienen muchas maneras de construirse, para eso se construyeron en total 70 reglas de detección de núcleo y 184 reglas de clasificación de argumentos.

Las heurísticas de detección de núcleo tienen un 95,3% de precisión (98,7% si no se consideran los núcleos de las coordinaciones). Las heurísticas de clasificación de argumentos tienen en promedio un 92,5% de precisión.

7.1.3 Léxico HPSG

A partir del corpus se extrajeron las entradas léxicas de las distintas categorías gramaticales. Debido a que existen muchas entradas léxicas diferentes, se utilizó una técnica para agruparlas según sus propiedades sintáctico-semánticas en unidades que denominamos *frames léxicos*.

Los *frames léxicos* agrupan conjuntos de entradas léxicas que comparten las mismas propiedades combinatorias: toman los mismos argumentos sintácticos y semánticos. Los *frames léxicos* se construyeron y analizaron para las tres categorías léxicas con más variabilidad: verbos, nombres y adjetivos.

En total se construyeron 725 *frames léxicos* para verbos, 161 para nombres y 49 para adjetivos. Estos *frames léxicos* agrupan los lemas existentes en el corpus: 2.375 lemas de verbos, 18.818 de nombres y 4.653 de adjetivos.

7.1.4 Supertagger

Se realizaron experimentos construyendo *supertaggers* para las tres categorías léxicas estudiadas, utilizando modelos tipo MaxEnt y CRF y diferentes atributos de entrenamiento.

El mejor clasificador para verbos utiliza un modelo CRF y fue entrenado usando 7 palabras de `contexto` y un `umbral` de palabras que aparezcan 20 veces o más. Su *accuracy* sobre verbos es de 82,35% considerando las tres etiquetas más probables.

El mejor clasificador para nombres utiliza un modelo CRF y fue entrenado usando `contexto` 5 y `umbral` 10. Su *accuracy* sobre nombres es de 91,23% considerando las tres etiquetas más probables.

El mejor clasificador para adjetivos utiliza un modelo CRF y fue entrenado usando `contexto` 5 y `umbral` 10. Su *accuracy* sobre adjetivos es de 84,21% considerando las tres etiquetas más probables.

Finalmente se construyó un *supertagger* unificado que etiqueta los *frames léxicos* de las tres categorías gramaticales consideradas. Considerando las tres etiquetas más probables, este *supertagger* tiene un *accuracy* de 83,58% para verbos, 85,78% para nombres y 81,40% para adjetivos.

7.2 Trabajo a futuro

7.2.1 Manejo de clíticos

En el idioma español los clíticos pueden aparecer en sustitución de uno de los argumentos de un verbo o como redundancia duplicando el argumento. En el presente trabajo se simplificó el modelado de estos casos, no se intenta identificar de qué tipo de clítico se trata, solamente se los detecta y se les aplica una regla `clitic_head` que sirve para absorberlos y poder continuar con el análisis. Sin embargo, los clíticos contienen información importante acerca de la estructura argumental del verbo que con este análisis simplificado no puede explotarse.

Se debe realizar un análisis más profundo de las frases introducidas por la regla `clitic_head` de manera de identificar en qué casos se trata de un clítico duplicado (en estos casos no interesa la estructura argumental, salvo para chequear consistencia) y en qué casos actúan como argumento del verbo.

También puede resultar de interés realizar un análisis más profundo e intentar inferir la entidad a la que refiere el clítico en los casos en que no es un duplicado. Este punto está relacionado con el problema de resolución de correferencias.

7.2.2 Manejo de expresiones relativas

Las oraciones relativas actúan como modificadores de nombres y son introducidas por una expresión pronominal relativa (por ejemplo “que” o “en donde”). Esta expresión pronominal puede actuar como sujeto o como objeto de la oración relativa, pero además está coindizada con el nombre que modifica. En este trabajo se marca la introducción de oraciones relativas mediante la regla `head_rel` pero se pierde cualquier otra información acerca de la relativa que se haya marcado en AnCora.

Se debe realizar un procesamiento extra para recuperar los siguientes datos acerca de la expresión relativa:

- Con qué rol argumental está actuando la expresión pronominal relativa dentro de la oración relativa.
- Qué nombre o frase nominal está siendo modificada por la expresión relativa.

7.2.3 Categorías gramaticales faltantes

En el presente trabajo nos concentramos en analizar los *frames léxicos* y construir clasificadores para tres categorías gramaticales: verbos, nombres y adjetivos. Se seleccionaron estas tres categorías debido a que son las que presentan más variabilidad en el corpus en cuanto a su uso. La representación de las demás categorías queda en cierta medida implícita: cada entrada léxica sería su propio *frame léxico*, y no se las agrupa según sus propiedades combinatorias.

En los casos en que sea necesario, se debe investigar las formas de agrupar las otras categorías gramaticales en *frames léxicos* de manera similar a lo que se hizo para verbos, nombres y adjetivos.

7.2.4 Mejoras en supertagging

Si bien los resultados obtenidos con los experimentos de *supertagging* son alentadores, existe un gran espacio para mejora. Hay otros experimentos que se pueden realizar para intentar mejorar los valores obtenidos, por ejemplo:

- Entrenar el *supertagger* unificado sobre todas las variantes de *features* para obtener el *supertagger* con el mejor promedio (no solamente el mejor para verbos).
- Incluir información morfológica en los *features* de entrenamiento, por ejemplo: género, número y persona (en las palabras donde sea adecuado). Si con estos *features* el *supertagger* aprende a reconocer que en español existen ciertos tipos de concordancia, podría explotar esta información para mejorar la performance global.
- Entrenar clasificadores utilizando combinaciones de atributos. Los experimentos realizados hasta el momento utilizan los *features* por separado, pero por ejemplo en [34] se utilizan *features* combinadas. Algunos ejemplos de este tipo de *features* son: combinación de lemas y categorías gramaticales de diferentes palabras del contexto y n-gramas de las categorías gramaticales del contexto.

7.2.5 Construcción del parser

Los recursos generados y herramientas generados son parte de lo necesario para construir un *parser* HPSG estadístico para el español. El proceso para parsear una oración sería el siguiente:

- Aplicar un analizador morfológico para obtener los lemas y categorías gramaticales (atributos necesarios para aplicar el *supertagger*).
- Aplicar el *supertagger* para obtener los *frames léxicos* más probables de cada palabra.
- Utilizar los *frames léxicos* y la información morfológica para construir las entradas léxicas más probables de cada palabra.

- Utilizar un algoritmo de *parsing* probabilístico para evaluar la probabilidad de combinar las diferentes entradas léxicas hasta formar oraciones.

Este último paso es el *parsing* en sí. Como nuestra gramática está definida en base a reglas binarias, es posible utilizar un algoritmo de *parsing* como el de CKY para obtener todos los posibles análisis y calcular sus probabilidades. La cantidad de análisis candidatos estará restringida debido a que el *supertagger* no devuelve todos los posibles *frames léxicos* para cada palabra sino solamente los más probables. De esta manera esperamos que el problema se vuelva tratable computacionalmente.

7.2.6 Reutilización de recursos para otros formalismos

Las gramáticas HPSG comparten características con otros formalismos gramaticales ricos. Es posible reutilizar parte del conocimiento producido durante el proceso de transformación del corpus a otros formalismos, como por ejemplo CCG. Los *frames léxicos* construidos pueden servir como base para crear entradas léxicas para CCG. La información sobre la identificación de núcleos sintácticos de los constituyentes y la clasificación de los argumentos resultará útil para intentar aplicar conceptos similares a una gramática de este tipo. El principal inconveniente en este caso sería que para CCG es importante conocer si los argumentos están ubicados a la derecha o a la izquierda del núcleo, ya que la forma que toma la categoría sintáctica del constituyente dependerá de esos factores.

7.2.7 Segmentación en cláusulas

Otro enfoque interesante que se puede investigar incluye el uso de un segmentador en cláusulas, por ejemplo la herramienta ClaTex [63]. ClaTex es un segmentador de proposiciones para el español. La noción de proposición que se utiliza en este sistema tiene ciertos puntos de contacto con la noción de "árbol HPSG elemental" que utilizamos en el presente trabajo. Las proposiciones de ClaTex son segmentos de texto que contienen un verbo conjugado con sus argumentos y modificadores. Por lo tanto, el árbol de *parsing* de una de estas proposiciones se corresponde con uno de los árboles elementales. ClaTex segmenta las oraciones y establece una estructura entre las proposiciones detectadas, por lo que sería interesante investigar si es posible utilizar la información de esta segmentación como base para un análisis sintáctico profundo: las proposiciones se analizan como árboles elementales mientras que la relación entre proposiciones provee información para armar las estructuras de mayor orden.

Bibliografía

[1] M. Collins, "Three generative, lexicalised models for statistical parsing," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 16–23, Association for Computational Linguistics, 1997.

[2] R. Tsarfaty, D. Seddah, S. Kübler, and J. Nivre, "Parsing morphologically rich languages: Introduction to the special issue," *Computational Linguistics*, vol. 39, no. 1, pp. 15–22, 2013.

[3] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 1, pp. 740–750, 2014.

[4] M. Steedman, "A very short introduction to ccg," *Unpublished paper. http://www. coqsci. ed. ac. uk/steedman/paper. html*, 1996.

[5] S. Clark and J. Hockenmaier, "Evaluating a wide-coverage ccg parser," in *Proceedings of the LREC 2002 Beyond PARSEVAL workshop*, pp. 60–66, Citeseer, 2002.

[6] A. K. Joshi, "Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?," 1985.

[7] A. K. Joshi and Y. Schabes, "Tree-adjoining grammars," in *Handbook of formal languages*, pp. 69–123, Springer, 1997.

[8] D. Chiang, "Statistical parsing with an automatically-extracted tree adjoining grammar," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 456–463, Association for Computational Linguistics, 2000.

[9] D. J. Weir and A. K. Joshi, "Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems," in *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pp. 278–285, Association for Computational Linguistics, 1988.

[10] A. K. Joshi, "Mildly context-sensitive grammars," *Oxford International Encyclopedia of Linguistics, 2nd Edition. Oxford University Press, Oxford, UK*, 2003.

[11] M. Lloberes, I. Castellón, and L. Padró, "Spanish freeling dependency grammar.," in *LREC*, vol. 10, pp. 693–699, 2010.

[12] L. Padró, M. Collado, S. Reese, M. Lloberes, I. Castellón, *et al.*, "Freeling 2.1: Five years of open-source language processing tools," 2010.

[13] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi, "Maltparser: A language-independent system for data-driven dependency parsing," *Natural Language Engineering*, vol. 13, no. 02, pp. 95–135, 2007.

[14] M. Ballesteros and J. Nivre, "Maltoptimizer: A system for maltparser optimization.," in *LREC*, pp. 2757–2763, 2012.

[15] C. Pollard and I. A. Sag, *Head-driven phrase structure grammar*. University of Chicago Press, 1994.

[16] B. Carpenter, *The logic of typed feature structures*. Cambridge University Press, 1992.

[17] N. Chomsky, "Remarks on nominalization," 1968.

[18] N. Chomsky, "Bare phrase structure," *Evolution and revolution in linguistic theory*, pp. 51–109, 1995.

[19] B. Carpenter, "The generative power of categorial grammars and head-driven phrase structure grammars with lexical rules," *Computational linguistics*, vol. 17, no. 3, pp. 301–313, 1991.

[20] "Delph-in." `http://www.delph-in.net/wiki/index.php/Home`. Acceso: 2015-10-04.

[21] F. Bond, S. Oepen, M. Siegel, A. Copestake, and D. Flickinger, "Open source machine translation with delph-in.," 2005.

[22] E. M. Bender, D. Flickinger, and S. Oepen, "The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars," in *Proceedings of the 2002 workshop on Grammar engineering and evaluation-Volume 15*, pp. 1–7, Association for Computational Linguistics, 2002.

[23] A. Copestake, J. Carroll, R. Malouf, and S. Oepen, "The (new) lkb system," *Center for the Study of Language and Information, Stanford University*, 1999.

[24] U. Callmeier, "Pet–a platform for experimentation with efficient hpsg processing techniques," *Natural Language Engineering*, vol. 6, no. 01, pp. 99–107, 2000.

[25] M. Marimon, "The spanish resource grammar.," in *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, pp. 17–23, 2010.

[26] M. Marimon, N. Bel, S. Espeja, and N. Seghezzi, "The spanish resource grammar: pre-processing strategy and lexical acquisition," in *Proceedings of the Workshop on Deep Linguistic Processing*, pp. 105–111, Association for Computational Linguistics, 2007.

[27] A. Copestake, D. Flickinger, C. Pollard, and I. A. Sag, "Minimal recursion semantics: An introduction," *Research on Language and Computation*, vol. 3, no. 2-3, pp. 281–332, 2005.

[28] M. Marimon, N. Bel, and L. Padró, "Automatic selection of hpsg-parsed sentences for treebank construction," *Computational Linguistics*, vol. 40, no. 3, pp. 523–531, 2014.

[29] M. Marimon, N. Bel, B. Fisas, B. Arias, S. Vázquez, J. Vivaldi, C. Morell, and M. Lorente, "The iula spanish lsp treebank," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, (Reykjavik, Iceland), May 2014.

[30] T. Ninomiya, T. Matsuzaki, Y. Tsuruoka, Y. Miyao, and J. Tsujii, "Extremely lexicalized models for accurate and fast hpsg parsing," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 155–163, Association for Computational Linguistics, 2006.

[31] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[32] Y. Miyao, T. Ninomiya, and J. Tsujii, "Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank," in *Natural Language Processing–IJCNLP 2004*, pp. 684–693, Springer, 2005.

[33] Y. Miyao and J. Tsujii, "Probabilistic disambiguation models for wide-coverage hpsg parsing," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 83–90, Association for Computational Linguistics, 2005.

[34] T. Matsuzaki, Y. Miyao, and J. Tsujii, "Efficient hpsg parsing with supertagging and cfg-filtering.," in *IJCAI*, pp. 1671–1676, 2007.

[35] A. K. Joshi and B. Srinivas, "Disambiguation of super parts of speech (or supertags): Almost parsing," in *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pp. 154–160, Association for Computational Linguistics, 1994.

[36] Y. Zhang, T. Matsuzaki, and J. Tsujii, "Forest-guided supertagger training," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 1281–1289, Association for Computational Linguistics, 2010.

[37] M. Takaki, Y. Minoru, T. Kentaro, and T. Jun'ichi, "Lilfes: towards a practical hpsg parser," in *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pp. 807–811, Association for Computational Linguistics, 1998.

[38] P. Kingsbury and M. Palmer, "From treebank to propbank.," in *LREC*, Citeseer, 2002.

[39] M. Taulé, M. A. Martí, and M. Recasens, "Ancora: Multilevel annotated corpora for catalan and spanish.," in *LREC*, 2008.

[40] M. A. Martí, M. Taulé, M. Bertran, and L. Màrquez, "Ancora: Multilingual and multilevel annotated corpora," *MS, Universitat de Barcelona*, 2007.

[41] I. A. Sag, T. Wasow, and E. M. Bender, "Syntactic theory: A formal introduction," 2003.

[42] M. Palmer, D. Gildea, and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Computational linguistics*, vol. 31, no. 1, pp. 71–106, 2005.

[43] L. Pineda and I. Meza, "Una gramática básica del español en hpsg," *Reporte interno, Depto. De Ciencias de la Computación, IIMAS, UNAM, Mex*, 2003.

[44] "Introducción a las etiquetas eagles." `http://www.cs.upc.edu/~nlp/tools/parole-sp.html`. Acceso: 2015-09-24.

[45] J. Hockenmaier, "Data and models for statistical parsing with combinatory categorial grammar," 2003.

[46] Y. Miyao, *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. PhD thesis, 2006.

[47] M. Collins, "Head-driven statistical models for natural language parsing," *Computational linguistics*, vol. 29, no. 4, pp. 589–637, 2003.

[48] "Documentación de ancora: Sintaxis (constituyentes)." `http://clic.ub.edu/corpus/webfm_send/20`. Acceso: 2015-09-16.

[49] L. Pineda and I. Meza, "The spanish pronominal clitic system," *Procesamiento del lenguaje natural*, vol. 34, pp. 67–103, 2005.

[50] L. Chiruzzo and D. Wonsever, "Desarrollo de un parser hpsg estadístico para el español," in *Proceedings of I Workshop on Tools and Resources for Automatically Processing Portuguese and Spanish*, (São Carlos, SP, Brazil), 2014.

[51] I. Castellón, A. Fernández-Montraveta, G. Vázquez, L. Alonso, and J. Capilla, "The sensem corpus: a corpus annotated at the syntactic and semantic level," in *5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006.

[52] L. Alonso, J. A. Capilla, I. Castellón, A. Fernández-Montraveta, and G. Vázquez, "The sensem project: Syntactico-semantic annotation of sentences in spanish," *Amsterdam Studies in the Theory and History of Linguistic Science Series 4*, vol. 292, p. 89, 2007.

[53] E. E. Ferrer, "Towards a semantic classification of spanish verbs based on subcategorisation information," in *Proceedings of the ACL 2004 workshop on Student research*, p. 13, Association for Computational Linguistics, 2004.

[54] A. Di Tullio and M. Malcouri, "Gramática del español para maestros y profesores del uruguay," 2013.

[55] J. R. Curran, S. Clark, and D. Vadas, "Multi-tagging for lexicalized-grammar parsing," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 697–704, Association for Computational Linguistics, 2006.

[56] R. Dridan, *Using lexical statistics to improve HPSG parsing*. PhD thesis, University of Saarland, 2009.

[57] B. Kiefer and H.-U. Krieger, "A context-free approximation of head-driven phrase structure grammar," in *Sixth International Workshop on Parsing Technologies*, 2000.

[58] J. Silva and A. Branco, "Assigning deep lexical types," in *Text, Speech and Dialogue*, pp. 240–247, Springer, 2012.

[59] P. Kolachina, S. Bangalore, and S. Kolachina, "Extracting ltag grammars from a spanish treebank," 2011.

[60] L. Chiruzzo and D. Wonsever, "Supertagging for a statistical hpsg parser for spanish," in *Statistical Language and Speech Processing*, pp. 18–26, Springer, 2015.

[61] C. Manning and D. Klein, "Stanford classifier." Software available at `http://nlp.stanford.edu/software/classifier.shtml`, 2003.

[62] T. Kudo, "Crf++: Yet another crf toolkit." Software available at `http://crfpp.sourceforge.net`, 2005.

[63] 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.

Anexos

Anexo A

Reglas de transformación del corpus

- `sp/arg=arg0∧!adjunct=yes`
  `grup.nom31`: [[$_{H}$ acuerdo] [de venta de energía] [$_{C}$ de EAA] [con la Comisión_Federal_de_Electricidad (CFE)]]
- `sp/arg=arg1∧!adjunct=yes`
  `grup.nom5`: [[$_{H}$ compra] [$_{C}$ del 51_por_ciento de la empresa mexicana (...)]]
- `sp/arg=arg2∧!adjunct=yes`
  `grup.nom37`: [[$_{H}$ duración] [$_{C}$ de 25 años]]
- `sp/arg=arg3∧!adjunct=yes`
  `grup.nom4436`: [[$_{H}$ ampliación] [$_{C}$ de ocho] [a doce meses] [del periodo de referencia (...)]]
- `sp/arg=arg4∧!adjunct=yes`
  `grup.nom5344`: [[única] [$_{H}$ salida] [$_{C}$ al exterior]]
- `sp/func=cn∧arg=arg0`
  `grup.nom10428`: [[$_{H}$ gestos y pasos "] [$_{C}$ del PP]]
- `sp/func=cn∧arg=arg1`
  `grup.nom11272`: [[$_{H}$ ampliación y modernización] [$_{C}$ de la Autopista_del_Oeste (...)]]
- `sp/func=cn∧arg=argL`
  `grup.nom25348`: [[$_{H}$ conferencia] [$_{C}$ de prensa]]
- `S/func=cn∧arg=arg1`
  `grup.nom38466`: [[$_{H}$ acciones] [$_{C}$ que puedan obstaculizar los procesos nacionales]]

Tabla A.1: Reglas completas para identificación de complementos de un `grup.nom`

- `sp/!arg`
  `grup.nom24`: [[H combustible] [principal] [M en una central de ciclo combinado (...)]]
- `sp/arg=argM`
  `grup.nom102`: [[H intervención] [ante militantes populares] [M en el hotel de la capital malagueña (...)]]
- `sp/adjunct=yes`
  `grup.nom56166`: [[H regulación , indemnización y coste] [M para la empresa]]
- `s.a`
  `grup.nom56173`: [["] [H abuso] ["] [M empresarial] [en la utilización de ese tipo de contratación]]
- `sa`
  `grup.nom108943`: [[H miembro] [M Scout] [honorario] [en Ecuador]]
- `a`
  `grup.nom5824`: [[M ex] [H vicepresidente]]
- `grup.a`
  `grup.nom26614`: [[H Todo] [lo] [M contrario]]
- `participi`
  `grup.nom56409`: [[M elaborados] [H cárnicos]]
- `S`
  `grup.nom56413`: [[H grupo] [M presidido por Ballvé]]
- `sn`
  `grup.nom56426`: [[H grupo] [M Campofrío] [, que registró durante 1999 unas (...)]]
- `grup.nom`
  `grup.nom113933`: [[H demócrata] [M Bill_Bradley , que también retiró su candidatura presidencial ,]]
- `n`
  `grup.nom10082`: [["] [H papas] [M fritas] ["]]
- `sadv/func=cn`
  `grup.nom37320`: [[H retorno] [a los cuadriláteros] [M después de haber perdido el 18_de_septiembre frente a Trinidad]]
- `sadv`
  `grup.nom37482`: [[H minutos] [M más] [que se jugaron en el estadio San_Carlos_de_Apoquindo , en Santiago]]
- `neg`
  `grup.nom52398`: [[M no] [H intervención]]
- `block`
  `grup.nom67697`: [["] [H alfarero] [de la interpretación] [M (. . .)] [capaz de moldearse como arcilla por fuera y (. . .)] ["]] *Este es un ejemplo en que se toma como modificador una estructura que fue transformada expresamente en la etapa top-down. Evita que sea una secuencia de puntuaciones.*
- `*/id=c14053` y `*/id=c14054`
  Estas son excepciones para simplificar el tratamiento de `grup.nom74824`: [[M ni] [H Sabas] [M ni] [Javi_Guerrero]]

Tabla A.2: Reglas completas para identificación de modificadores de un `grup.nom`

- `grup.verb`
  `sentence1`: [[El grupo estatal Electricité_de_France (EDF)] [$_{H}$anunció] [hoy , jueves ,] [la compra del 51_por_ciento de la empresa mexicana (...)] [.]]
- `participi`
  `S28`: [[$_{H}$elegido] [como sede de la noche electoral del PP]]
- `infinitiu`
  `S37`: [[$_{H}$llevar] [al PP] [hasta la victoria]]
- `gerundi`
  `S421`: [[$_{H}$llevando] [en la mano] [un balón hinchable que representaba (. . .)]]
- `v`
  `S2499`: [[$_{H}$comprada] [por Entergy_Londres]]
- `S`
  `S40`: [[$_{H}$ilusionar y convencer] [a muchos] [para que apuesten " por este partido]]
- `sentence`
  `S37`: [[$_{H}$llevar] [al PP] [hasta la victoria]]
- `sn`
  `S210`: [[$_{H}$la segunda , mucho más potente ,] [a las 07.30.42 GMT]]
- `sa`
  `S4409`: [[( ] [$_{H}$segundo] [ayer] [, con 8,33] [ )]]
- `sp`
  `S97`: [[$_{H}$a Uzbekistán] [con Holanda]]
- `interjeccio`
  `sentence9131`: Originalmente la oración es: [[-] [-] [¡] [Vaya] [debut] [!] [-] [-] [.]] Luego de la transformación *top-down*, estos componentes se simplifican y lo que queda para procesar es: [[-] [-] [$_{H}$¡ Vaya debut !] [-] [-] [.]]
- `neg`
  `sentence9656`: [[-] [-] [Siempre] [$_{H}$no] [.]]
- `block`
  `sentence4847`: Originalmente la oración es: [[Por_ejemplo ,] [¿] [puede un informe (. . .) ser] [objeto de una nueva apelación] [?] [.]] Luego de la transformación top-down, estos componentes se simplifican y lo que queda es: [[Por_ejemplo ,] [$_{H}$¿ puede un informe (. . .) ser objeto de una nueva apelación ?] [.]]
- `sadv`
  `S16550`: [[$_{H}$ahora] [más que nunca]]

Tabla A.3: Reglas completas para identificación de núcleo de una oración

- `sn/func=cd` — `S2`: [[$_{H}$poner_en_marcha] [$_{C}$una central de gas de 495 megavatios]]
- `sn/func=atr` — `S121`: [[una de bronce] [no] [$_{H}$sería] [para mí] [$_{C}$una decepción]]
- `sn/func=cpred` — `S157`: [[$_{H}$denominado] [$_{C}$" cinturón rojo "]]
- `sn/func=ci` — `sentence5886`: [[De esta manera ,] [$_{C}$los inversores que quieran comprar dólares] [le] [$_{H}$costaba] [cada billete verde] [168,28 pesetas , el nivel más alto desde 1985] [.]]
- `sn/func=cc^arg=arg2` — `S20451`: [[mientras_que] [la cifra de negocios] [$_{H}$aumentó] [$_{C}$un ocho_por_ciento] [, hasta los 44.350 millones de pesetas (. . .)]]
- `S/func=cd` — `sentence6542`: [[Por eso ,] [$_{H}$opinó] [$_{C}$que las listas de espera no son una " enfermedad " del sistema sanitario (. . .)]]
- `S/func=cpred` — `S20747`: [[que] ["] [$_{H}$ha quedado] [$_{C}$confirmado] [por boca del propio señor Toledo] ["] [que Berenson podría " beneficiarse de una serie de medidas (. . .)]]
- `S/func=atr` — `S20775`: [[Todos los terroristas peruanos y extranjeros que fueron juzgados en el fuero militar] [$_{H}$están] [$_{C}$perfectamente bien condenados]]
- `sp/func=cpred` — `sentence6684`: [[Según el alcalde de Jerez_de_la_Frontera y candidato andalucista] [, tras el acuerdo federal entre el PSOE e IU ,] [el PA] ["] [se] [$_{H}$quedó] [casi en exclusiva] [$_{C}$como opción de centro izquierda " , " tranquilo " y " en_medio_de la derecha de siempre y la izquierda "] [.]]
- `sp/func=creg` — `S21067`: [[que] [Andalucía] [$_{H}$converja] [$_{C}$con Europa]]
- `sp/func=cd` — `S21078`: [[El cabeza de lista del PA (...)] [$_{H}$advirtió] [$_{C}$al presidente de la Generalitat , Jordi_Pujol ,] [de que los andalucistas no aceptarán que existan " singularidades económicas " (. . .)]]
- `sp/func=cag` — `S21087`: [[$_{H}$firmado] [$_{C}$por el PA] [con UAGA] [- que ya lo hizo antes con el PSOE e IU -]]
- `sp/arg=arg1` — `S21105`: [[$_{H}$llegar] [a la una y media de la madrugada] [$_{C}$a un nutrido grupo de encapuchados]]
- `sp/arg=arg2` — `sentence6706`: [[La Ertzaintza , que acudió minutos después ,] [$_{H}$recogió] [$_{C}$del suelo] [casquillos del calibre 7,65 utilizados en los fusiles Cetme y de 9 milímetros] [.]]
- `sp/arg=arg3` — `S21222`: [[que] [este conflicto] [$_{H}$comporte] [un frenazo] [$_{C}$a cualquier pretensión de unificar más que hasta ahora a los países de la UE]] Esto parece un error, no es un complemento del verbo.
- `sp/arg=arg4` — `sentence6776`: [[El expreso de la compañía GNER] [$_{H}$había salido] [de Newcastle] [a las 4.45 de la madrugada] [$_{C}$con destino a la estación de Charing_Cross , en Londres , donde tenía prevista la llegada a las ocho de la mañana] [.]]
- `sa/func=cpred` — `S21390`: [[que] [IU] [no] [$_{H}$consideraba] ["] [$_{C}$coherente] [la propuesta de retirada de sus candidaturas] [, porque va en_detrimento_de una de las fuerzas de la izquierda al no permitir mantener su identidad en todo el Estado] ["]]
- `sa/func=atr` — `sentence6829`: [[El holandés Richard_Krajicek , uno de los beneficiados (...)] [$_{H}$fue] [$_{C}$más imparcial] [en este sentido] [al comentar la actitud de Coretja y Costa] [.]]
- `sa/func=atr` — `sentence6829`: [[El holandés Richard_Krajicek , uno de los beneficiados (...)] [$_{H}$fue] [$_{C}$más imparcial] [en este sentido] [al comentar la actitud de Coretja y Costa] [.]]
- `sadv/func=atr` — `sentence6829`: [[ ] [no] [$_{H}$es] [$_{C}$así]]
- `sadv/func=cd` — `sentence7184`: [[-] [-] [Una imagen] [$_{H}$vale] [$_{C}$más que mil palabras] [.]]
- `sadv/func=cc^arg=arg2` — `S23298`: [[que] [su entrenador] [$_{H}$pudiera estar] [$_{C}$cerca de ella] [durante los combates]]
- `sadv/func=cc^arg=arg4` — `sentence10852`: [["] [En 1924 ,] [ ] [$_{H}$subió] [$_{C}$a_bordo del primer convoy , en cuya fabricación participó] ["] [.]]
- `relatiu/func=cd` — `S12686`: [[ ] [$_{C}$que] [$_{H}$ha recuperado] [este año del Atalanta]] Este caso es raro, no se identifica bien el rol del "que" como expresión pronominal relativa debido a que hay un sintagma nominal vacío (elidido) al inicio del constituyente.
- `S/clausetype=completive` — `S12876`: [[no] [le] [$_{H}$corresponde] [$_{C}$tomar esa determinación] [a ninguno de los candidatos , ni a los observadores ,]]

Tabla A.4: Reglas completas para identificación de complementos de una oración

Anexo B

Evaluación de la transformación

Categoría del padre	Total	Etiquetadas correctamente
grup.a	7	4
grup.adv	3	1
grup.nom	238	195
grup.verb	18	18
infinitiu	3	3
relatiu	1	1
S	214	189
s.a	3	2
sa	1	1
sadv	12	11
sentence	151	144
sn	29	28
sp	228	227
spec	8	8

Tabla B.1: Desglose de los elementos clasificados (para cualquier relación) agrupados por categoría del constituyente padre.

Categoría del padre	Total	Etiquetadas correctamente
grup.a	3	0
grup.adv	2	0
grup.nom	58	23
grup.verb	18	18
infinitiu	3	3
S	73	59
sadv	1	0
sentence	31	27
sp	203	203

Tabla B.2: Desglose de los elementos clasificados con la relación "complemento" agrupados por categoría del constituyente padre.

Anexo C

Performance de los experimentos

C.1 Verbos

Los valores de perfomance son:
[*accuracy* para todas las categorías] | [*accuracy* solamente para verbos]

C.1.1 Experimentos con lemas

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
10	5	88.21 | 36.52	93.26 | 53.57	95.13 | 63.25	96.29 | 69.59	96.99 | 74.08
10	7	87.79 | 35.20	93.00 | 53.17	94.99 | 62.79	96.11 | 68.32	96.82 | 72.81
10	9	87.31 | 33.58	92.56 | 50.98	94.72 | 60.83	95.90 | 67.40	96.76 | 72.52
15	5	89.10 | 37.04	93.64 | 54.38	95.37 | 64.00	96.54 | 70.62	97.18 | 74.88
20	5	89.70 | 37.38	93.92 | 54.72	95.57 | 64.23	96.61 | 70.56	97.28 | 75.06
20	7	89.27 | 36.29	93.71 | 54.49	95.49 | 64.40	96.50 | 69.82	97.14 | 74.31
20	9	88.96 | 35.02	93.37 | 52.30	95.27 | 62.79	96.30 | 68.89	97.00 | 73.73
20	11	88.61 | 34.56	93.08 | 50.40	95.12 | 61.58	96.14 | 67.22	96.89 | 72.64
30	5	90.43 | 38.48	94.22 | 55.36	95.86 | 65.73	96.87 | 71.95	97.43 | 76.27
30	7	89.90 | 36.81	94.01 | 55.41	95.72 | 64.80	96.70 | 71.14	97.33 | 75.75
30	9	89.63 | 35.89	93.84 | 54.09	95.58 | 63.88	96.52 | 70.05	97.21 | 74.31
40	5	90.93 | 38.31	94.49 | 55.93	96.03 | 66.71	96.93 | 72.70	97.52 | 77.30
50	5	91.28 | 38.25	94.74 | 56.22	96.17 | 67.28	97.02 | 72.41	97.66 | 77.65

Tabla C.1: Performance CRF para los verbos entrenando solo con lemas, corpus de 50.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
20	7	91.05 | 41.99	94.98 | 61.23	96.38 | 70.79	97.25 | 76.96	97.78 | 80.24
20	9	90.90 | 41.94	94.74 | 60.31	96.31 | 69.87	97.19 | 76.27	97.72 | 79.84
30	7	91.49 | 42.51	95.21 | 61.87	96.54 | 71.60	97.39 | 77.42	97.88 | 80.65
30	9	91.34 | 42.05	95.04 | 60.94	96.42 | 69.99	97.25 | 76.09	97.79 | 80.01
50	5	92.41 | 43.78	95.47 | 61.23	96.82 | 72.87	97.61 | 77.76	98.10 | 81.74
50	11	91.77 | 40.84	95.04 | 59.27	96.52 | 70.33	97.30 | 76.21	97.88 | 80.24

Tabla C.2: Performance CRF para los verbos entrenando solo con lemas, corpus de 106.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	5	93.15 | 48.79	96.37 | 67.57	97.67 | 78.05	98.43 | 83.58	98.81 | 86.75
50	5	93.34 | 48.33	96.35 | 66.94	97.65 | 77.36	98.40 | 82.95	98.80 | 86.69

Tabla C.3: Performance CRF para los verbos entrenando solo con lemas, corpus completo (470.000 palabras).

C.1.2 Experimentos con lemas y POS

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	5	93.59 | 41.95	95.93 | 62.72	97.09 | 72.70	97.71 | 77.95	98.18 | 81.86
30	7	93.68 | 42.24	95.81 | 62.37	96.94 | 72.29	97.68 | 78.30	98.15 | 81.86
30	9	93.62 | 40.84	95.73 | 61.20	96.96 | 71.88	97.66 | 77.83	98.15 | 81.04
30	11	93.53 | 39.96	95.60 | 59.33	96.86 | 70.48	97.51 | 75.96	98.02 | 79.58
50	5	93.56 | 41.13	95.81 | 61.09	96.98 | 71.88	97.75 | 78.00	98.13 | 81.33
50	7	93.65 | 42.53	95.77 | 61.84	96.98 | 72.64	97.66 | 77.89	98.16 | 82.21
50	9	93.61 | 40.72	95.80 | 61.32	96.95 | 71.41	97.61 | 76.90	98.15 | 80.92
50	11	93.49 | 40.02	95.62 | 59.22	96.76 | 69.72	97.50 | 75.90	98.03 | 79.93

Tabla C.4: Performance CRF para los verbos entrenando lemas y POS, corpus de 50.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	5	94.05 | 47.43	96.13 | 64.64	97.38 | 75.90	98.11 | 81.80	98.48 | 84.71
30	7	93.92 | 45.92	96.17 | 65.75	97.39 | 76.43	98.09 | 81.80	98.51 | 85.18
30	9	93.96 | 46.27	96.10 | 66.16	97.37 | 77.07	98.06 | 82.03	98.48 | 84.71
30	11	93.92 | 45.97	96.04 | 64.59	97.21 | 74.97	97.95 | 81.04	98.41 | 84.54
50	5	93.98 | 46.38	96.03 | 63.77	97.31 | 75.20	98.06 | 81.33	98.48 | 84.71
50	7	93.94 | 46.32	96.12 | 65.75	97.27 | 75.85	98.02 | 81.45	98.50 | 85.12
50	9	93.93 | 45.86	96.04 | 65.64	97.27 | 75.73	98.03 | 81.97	98.45 | 84.66
50	11	93.87 | 45.45	96.04 | 64.99	97.21 | 74.91	97.87 | 80.57	98.31 | 83.55

Tabla C.5: Performance CRF para los verbos entrenando lemas y POS, corpus de 106.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	7	94.33 | 50.93	96.49 | 69.49	97.72 | 80.11	98.41 | 85.18	98.74 | 87.57
30	9	94.22 | 49.94	96.48 | 69.02	97.69 | 80.11	98.33 | 84.95	98.77 | 87.98
50	7	94.29 | 50.29	96.41 | 69.02	97.68 | 79.70	98.35 | 84.42	98.72 | 87.51
50	9	94.19 | 49.47	96.36 | 68.2	97.61 | 79.23	98.28 | 84.42	98.74 | 87.69

Tabla C.6: Performance CRF para los verbos entrenando lemas y POS, corpus de 218.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	7	94.57 | 51.75	96.9 | 71.41	98.11 | 81.51	98.80 | 87.51	99.12 | 90.20
30	9	94.50 | 51.52	96.84 | 70.65	98.09 | 81.21	98.73 | 86.81	99.14 | 90.49
50	7	94.51 | 51.34	96.82 | 70.30	98.03 | 80.34	98.69 | 86.11	99.05 | 89.44

Tabla C.7: Performance CRF para los verbos entrenando lemas y POS, corpus completo (470.000 palabras).

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	5	94.03 | 45.59	96.32 | 66.49	97.68 | 78.90	98.37 | 85.17	98.77 | 88.79
30	7	94.03 | 45.59	96.36 | 66.81	97.64 | 78.54	98.36 | 85.06	98.72 | 88.31
30	9	94.02 | 45.55	96.23 | 65.62	97.59 | 78.06	98.28 | 84.30	98.73 | 88.43
30	11	93.95 | 44.87	96.21 | 65.50	97.54 | 77.62	98.19 | 83.51	98.61 | 87.32

Tabla C.8: Performance MaxEnt para los verbos entrenando lemas y POS, corpus completo (470.000 palabras).

C.1.3 Experimentos con oraciones que tienen todos los argumentos marcados

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
20	5	94.97 | 50.82	96.75 | 68.24	97.62 | 76.71	98.04 | 80.86	98.40 | 84.39
20	7	94.84 | 49.57	96.82 | 68.86	97.62 | 76.71	98.05 | 80.94	98.36 | 84.00
20	9	94.78 | 48.94	96.61 | 66.82	97.52 | 75.76	98.05 | 80.94	98.36 | 84.00
20	11	94.56 | 46.75	96.49 | 65.65	97.38 | 74.35	97.88 | 79.29	98.24 | 82.82
30	5	94.92 | 50.35	96.77 | 68.39	97.59 | 76.39	98.01 | 80.55	98.40 | 84.39
30	7	94.82 | 49.33	96.78 | 68.47	97.56 | 76.16	98.10 | 81.41	98.39 | 84.24
30	9	94.65 | 47.69	96.65 | 67.22	97.59 | 76.47	98.00 | 80.39	98.34 | 83.76
30	11	94.57 | 46.90	96.46 | 65.33	97.35 | 74.04	97.93 | 79.76	98.25 | 82.90
50	5	94.92 | 50.27	96.74 | 68.08	97.57 | 76.24	98.03 | 80.71	98.32 | 83.61
50	7	94.72 | 48.39	96.76 | 68.31	97.55 | 76.08	98.04 | 80.86	98.30 | 83.37
50	9	94.72 | 48.39	96.62 | 66.90	97.57 | 76.24	97.97 | 80.16	98.29 | 83.29

Tabla C.9: Performance CRF para los verbos entrenado con los que tienen argumentos, corpus de 60.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
20	5	95.24 | 53.49	97.17 | 72.31	97.83 | 78.82	98.25 | 82.90	98.61 | 86.43
20	7	95.26 | 53.65	97.15 | 72.08	98.00 | 80.39	98.30 | 83.37	98.62 | 86.51
20	9	95.16 | 52.63	97.11 | 71.76	97.86 | 79.06	98.28 | 83.14	98.60 | 86.35
20	11	95.11 | 52.16	97.02 | 70.82	97.80 | 78.51	98.24 | 82.82	98.56 | 85.88
30	5	95.23 | 53.33	97.18 | 72.39	97.81 | 78.59	98.22 | 82.59	98.56 | 85.88
30	7	95.14 | 52.47	97.15 | 72.08	97.98 | 80.24	98.31 | 83.45	98.56 | 85.88
30	9	95.16 | 52.71	97.11 | 71.76	97.88 | 79.29	98.27 | 83.06	98.57 | 86.04
30	11	95.10 | 52.08	97.07 | 71.37	97.79 | 78.43	98.27 | 83.06	98.51 | 85.41

Tabla C.10: Performance CRF para los verbos entrenado con los que tienen argumentos, corpus de 120.000 palabras.

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
20	7	95.64 | 57.33	97.53 | 75.84	98.20 | 82.35	98.64 | 86.67	98.85 | 88.78
30	7	95.63 | 57.25	97.53 | 75.84	98.15 | 81.88	98.61 | 86.43	98.85 | 88.71

Tabla C.11: Performance CRF para los verbos entrenado con los que tienen argumentos, corpus completo (250.000 palabras).

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
30	5	95.48 | 55.76	97.24 | 73.02	98.20 | 82.35	98.58 | 86.12	98.81 | 88.31
30	7	95.45 | 55.45	97.34 | 73.96	98.12 | 81.65	98.52 | 85.57	98.77 | 88.00
30	9	95.41 | 55.14	97.27 | 73.25	97.96 | 80.00	98.40 | 84.31	98.76 | 87.84
30	11	95.29 | 53.88	97.25 | 73.10	97.91 | 79.53	98.38 | 84.16	98.60 | 86.35

Tabla C.12: Performance MaxEnt para los verbos entrenado con los que tienen argumentos, corpus completo (250.000 palabras).

C.2 Nombres

C.2.1 Performance global de los experimentos

Los valores de perfomance son:
[*accuracy* para todas las categorías] | [*accuracy* solamente para nombres]

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
10	5	98.87 | 36.10	99.60 | 73.48	99.83 | 88.18	99.88 | 91.37	99.90 | 92.65
10	7	98.84 | 34.50	99.62 | 74.12	99.83 | 88.18	99.87 | 91.05	99.89 | 92.01
10	9	98.86 | 33.87	99.61 | 73.48	99.82 | 86.90	99.87 | 90.73	99.90 | 92.65
10	11	98.86 | 32.27	99.59 | 71.88	99.83 | 87.86	99.89 | 91.69	99.91 | 93.29
20	5	98.81 | 31.63	99.60 | 73.16	99.82 | 86.90	99.87 | 90.73	99.89 | 92.01
20	7	98.78 | 30.35	99.60 | 72.84	99.83 | 87.54	99.89 | 91.69	99.90 | 92.97
20	9	98.82 | 31.31	99.60 | 73.16	99.85 | 88.82	99.89 | 91.69	99.92 | 93.93
20	11	98.79 | 30.03	99.61 | 73.48	99.83 | 87.86	99.89 | 91.69	99.91 | 93.61
30	5	98.74 | 28.12	99.59 | 72.20	99.82 | 86.90	99.86 | 89.78	99.88 | 91.37
30	7	98.71 | 28.75	99.57 | 70.61	99.81 | 86.58	99.87 | 90.42	99.90 | 92.33
30	9	98.72 | 28.43	99.56 | 70.93	99.83 | 87.86	99.89 | 91.69	99.91 | 93.61
30	11	98.69 | 26.20	99.58 | 71.57	99.84 | 88.18	99.89 | 92.01	99.91 | 93.61

Tabla C.13: Performance CRF para los nombres, corpus completo (470.000 palabras).

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
10	5	98.80 | 40.26	99.55 | 71.57	99.79 | 85.62	99.87 | 91.05	99.92 | 94.25
10	7	98.79 | 39.30	99.52 | 69.65	99.81 | 86.58	99.87 | 90.73	99.89 | 92.33
10	9	98.84 | 38.66	99.58 | 72.84	99.82 | 87.54	99.87 | 90.73	99.90 | 92.33
10	11	98.86 | 39.30	99.56 | 70.61	99.83 | 88.18	99.88 | 91.69	99.91 | 93.61
20	5	98.75 | 34.82	99.53 | 69.65	99.78 | 84.35	99.87 | 91.05	99.91 | 93.61
20	7	98.68 | 32.59	99.48 | 66.77	99.78 | 84.03	99.87 | 90.73	99.89 | 92.33
20	9	98.74 | 34.19	99.51 | 68.69	99.81 | 86.58	99.89 | 91.69	99.91 | 93.61
20	11	98.73 | 34.50	99.53 | 69.01	99.80 | 86.90	99.88 | 91.37	99.92 | 94.25
30	5	98.71 | 31.95	99.50 | 67.73	99.77 | 84.03	99.88 | 91.37	99.90 | 92.65
30	7	98.58 | 30.35	99.42 | 63.90	99.78 | 84.35	99.86 | 89.78	99.89 | 92.01
30	9	98.54 | 31.31	99.50 | 68.05	99.76 | 83.71	99.86 | 89.78	99.90 | 92.65
30	11	98.60 | 31.95	99.51 | 68.37	99.76 | 83.39	99.87 | 91.05	99.92 | 93.93

Tabla C.14: Performance MaxEnt para los nombres, corpus completo (470.000 palabras).

C.2.2 Nombre genérico vs. nombre con complementos

Se muestra precisión y *recall* de los elementos marcados como subclases de nombre, considerando *hit* si una subclase es etiquetada como subclase (sin importar que sean diferentes) y *miss* si una subclase es etiquetada como nombre genérico.

Umbral	Contexto	Precisión	Recall	F1
10	5	0.6739	0.3962	0.2495
10	7	0.6685	0.3930	0.2475
10	9	0.6875	0.3866	0.2474
10	11	0.6933	0.3610	0.2374
20	5	0.6527	0.3482	0.2271
20	7	0.6353	0.3450	0.2236
20	9	0.6646	0.3482	0.2285
20	11	0.6463	0.3387	0.2222
30	5	0.6145	0.3259	0.2129
30	7	0.5978	0.3419	0.2175
30	9	0.6012	0.3323	0.2140
30	11	0.5808	0.3099	0.2021

Tabla C.15: Performance CRF: nombres genéricos vs. nombres con complementos.

Umbral	Contexto	Precisión	Recall	F1
10	5	0.6337	0.4920	0.2770
10	7	0.6364	0.4920	0.2775
10	9	0.6682	0.4760	0.2780
10	11	0.6667	0.4601	0.2722
20	5	0.6210	0.4345	0.2556
20	7	0.5893	0.4217	0.2458
20	9	0.6182	0.4345	0.2552
20	11	0.6081	0.4313	0.2523
30	5	0.6029	0.4026	0.2414
30	7	0.5408	0.4026	0.2308
30	9	0.5200	0.4153	0.2309
30	11	0.5401	0.4089	0.2327

Tabla C.16: Performance MaxEnt: nombres genéricos vs. nombres con complementos.

C.3 Adjetivos

Los valores de perfomance son:
[*accuracy* para todas las categorías] | [*accuracy* solamente para adjetivos]

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
10	5	99.79 | 66.25	99.95 | 86.25	99.97 | 90.00	99.97 | 92.50	99.98 | 95.00
10	7	99.76 | 58.75	99.94 | 86.25	99.96 | 88.75	99.97 | 92.50	99.98 | 93.75
10	9	99.75 | 56.25	99.95 | 86.25	99.97 | 90.00	99.98 | 93.75	99.98 | 93.75
10	11	99.76 | 57.50	99.94 | 83.75	99.97 | 90.00	99.98 | 93.75	99.98 | 93.75
20	5	99.79 | 62.50	99.94 | 85.00	99.97 | 90.00	99.97 | 92.50	99.98 | 95.00
20	7	99.75 | 57.50	99.94 | 85.00	99.97 | 91.25	99.98 | 95.00	99.81 | 95.00
20	9	99.71 | 51.25	99.94 | 85.00	99.96 | 88.75	99.97 | 92.50	99.98 | 93.75
20	11	99.74 | 55.00	99.94 | 83.75	99.96 | 87.50	99.97 | 92.50	99.98 | 93.75
30	5	99.77 | 58.75	99.93 | 85.00	99.96 | 87.50	99.97 | 91.25	99.98 | 93.75
30	7	99.72 | 52.50	99.93 | 85.00	99.96 | 87.50	99.97 | 92.50	99.98 | 95.00
30	9	99.70 | 51.25	99.94 | 85.00	99.95 | 86.25	99.97 | 91.25	99.98 | 93.75
30	11	99.73 | 53.75	99.93 | 83.75	99.96 | 87.50	99.98 | 93.75	99.99 | 96.25

Tabla C.17: Performance CRF para los adjetivos, corpus completo (470.000 palabras).

Umbral	Contexto	# etiquetas más frecuentes				
		1	2	3	4	5
10	5	99.76 | 65.00	99.94 | 85.00	99.97 | 91.25	99.98 | 93.75	99.98 | 95.00
10	7	99.74 | 57.50	99.94 | 86.25	99.97 | 90.00	99.97 | 92.50	99.98 | 93.75
10	9	99.73 | 56.25	99.95 | 86.25	99.97 | 91.25	99.98 | 93.75	99.98 | 93.75
10	11	99.75 | 57.50	99.94 | 85.00	99.97 | 91.25	99.97 | 92.50	99.98 | 93.75
20	5	99.74 | 60.00	99.93 | 85.00	99.97 | 92.50	99.99 | 97.50	99.99 | 97.50
20	7	99.75 | 58.75	99.93 | 83.75	99.97 | 91.25	99.98 | 95.00	99.99 | 97.50
20	9	99.72 | 53.75	99.93 | 82.50	99.97 | 90.00	99.97 | 92.50	99.98 | 95.00
20	11	99.73 | 55.00	99.94 | 83.75	99.97 | 90.00	99.98 | 93.75	99.98 | 95.00
30	5	99.72 | 58.75	99.93 | 82.50	99.97 | 91.25	99.99 | 96.25	100.0 | 98.75
30	7	99.70 | 51.25	99.93 | 83.75	99.97 | 91.25	99.98 | 95.00	99.99 | 96.25
30	9	99.69 | 53.75	99.93 | 82.50	99.96 | 88.75	99.97 | 91.25	99.98 | 93.75
30	11	99.71 | 53.75	99.92 | 81.25	99.96 | 88.75	99.98 | 95.00	99.99 | 96.25

Tabla C.18: Performance MaxEnt para los adjetivos, corpus completo (470.000 palabras).

C.4 Supertagger unificado

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	92.49	95.59	97.20	98.02	98.58
	solo verbos	54.61	69.97	79.39	85.09	88.45
	solo nombres	31.77	69.89	85.36	90.33	94.20
	solo adjetivos	53.93	82.02	92.13	96.63	98.88
corpus de test	todas las categorías	91.82	94.99	96.62	97.46	98.03
	solo verbos	53.72	68.74	77.59	82.29	85.49
	solo nombres	26.27	68.66	87.10	91.24	93.55
	solo adjetivos	44.83	68.97	74.14	86.21	86.21

Tabla C.19: *Accuracy* para el supertagger unificado: modelo MaxEnt, `umbral` 30, `contexto` 5, todos los verbos

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	93.95	96.70	97.75	98.23	98.58
	solo verbos	57.65	74.43	81.10	84.78	87.53
	solo nombres	33.97	64.59	83.73	89.47	93.30
	solo adjetivos	44.93	89.86	92.75	92.75	94.20
corpus de test	todas las categorías	94.20	96.87	97.83	98.39	98.77
	solo verbos	61.91	77.22	82.95	86.51	89.38
	solo nombres	33.82	67.65	84.31	93.14	96.08
	solo adjetivos	46.51	74.42	83.72	88.37	95.35

Tabla C.20: *Accuracy* para el supertagger unificado: modelo MaxEnt, `umbral` 20, `contexto` 7, solo verbos con argumentos

# etiquetas más frecuentes		1	2	3	4	5
corpus de desarrollo	todas las categorías	92.83	95.97	97.35	98.09	98.56
	solo verbos	57.81	73.23	80.87	85.34	88.70
	solo nombres	28.73	71.55	87.57	92.54	94.48
	solo adjetivos	47.19	85.39	88.76	93.26	93.26
corpus de test	todas las categorías	92.11	95.20	96.75	97.66	98.26
	solo verbos	54.78	70.87	78.66	83.79	87.35
	solo nombres	23.50	62.21	84.33	92.17	94.93
	solo adjetivos	39.66	68.97	81.03	84.48	87.93

Tabla C.21: *Accuracy* para el supertagger unificado: modelo CRF, `umbral` 30, `contexto` 7, todos los verbos