Proyecto de Fin de Máster de Sistemas Inteligentes

2009-2010

Mejora de la Precisión para el Análisis de Dependencias usando Maltparser para el Castellano.

Miguel Ballesteros Martínez

Ingeniero en Informática

Universidad Complutense de Madrid

Director del Trabajo:

Dr.Pablo Gervás

Asistido por:

Dra.Virginia Francisco y Jesús Herrera

Máster en Investigación en Informática
Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática, Universidad Complutense de Madrid

Mejora de la Precisión para el Análisis de Dependencias usando Maltparser para el Castellano.

Miguel Ballesteros Martínez

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática - UCM

21 de junio 2010

AGRADECIMIENTOS

Cuando no se piensa lo que se dice es cuando se dice lo que se piensa.
Jacinto Benavente.

A **Jesús Herrera** por ser, además de un fantástico guía en todo el proceso, un amigo. Puedo afirmar que este trabajo no habría sido posible sin su ayuda, constancia y cercanía.

A **Virginia Francisco** por enseñarme, apoyarme, servirme de referencia, exigirme y ser paciente a la vez.

A **Pablo Gervás** por darme la oportunidad, por su forma de ver las cosas, ayudarme, valorarme y animarme.

A **Susana Bautista** por orientarme, ayudarme e interesarse.

A **Sergio Valverde, Victor Requena, Juan Chazarra, Carlos Agra, Aleixa Helguera, Mónica Bazán, Carlos Sánchez y Milca Rosa** por escucharme, por los buenos momentos y por las risas.

A **Abel Miguez**, que ha estado un poco lejos este año, pero en realidad muy cerca. *Good Luck* my friend.

A **José Antonio Blanes**, que se le echa(mucho) de menos a la hora de trabajar pero gracias por las risas, las bromas, la cercanía e interés.

A **Alicia Martínez** por tener una paciencia infinita, por estar siempre, apoyarme y escucharme.

A **Jesús Ballesteros** por darme seguridad, ayudarme, enseñarme y por las risas de siempre.

A **Guillermo Ballesteros** por ser una referencia, explicarme y aconsejarme. *Grazie mille.*

A **María Fernández**, por todo, por cuidarme cuando menos podía, por quererme, buscarme cuando más hacía falta, por estar siempre, por ser *ella.*

Y a muchas otras personas que quizá no aparecen aquí pero que aunque no lo sepan su influencia este año ha sido muy importante en momentos puntuales y me han ayudado a ver las cosas de otra manera. Agustín, Raúl, Leticia, José Daniel, Raquel, Javier, Gonzalo, Alberto, Pablo, Federico, Fabio,

Lara, Basil, Philip, Rebecca, Abilio, Jesús, Alberto, Jaime, Javier, etc.

Gracias a todos.

Miguel Ballesteros Martínez.

Resumen en Castellano

Maltparser es un generador de analizadores de dependencias contemporáneo basado en aprendizaje automático mediante el que se obtiene una gran precisión. Actualmente los resultados rondan el 80 % de precisión (Labelled Attachment Score–LAS) para el castellano y parece que estos resultados son un límite que no se puede superar. En este trabajo se han realizado una serie de estudios sobre generadores de analizadores de dependencias en búsqueda de técnicas para mejorar este límite *de facto* de la precisión. En primer lugar, se muestran una serie de ideas y experimentos basados en el tamaño del corpus de entrenamiento y/o longitud de las frases, y en segundo lugar se aborda la idea fundamental de este trabajo: el Analizador de N-Versiones, que consiste en combinar la acción de diferentes analizadores entrenados específicamente para diversas tareas y conseguir de esta manera mejorar la calidad overall en el análisis de dependencias sintáctico.

Abstract in English

Maltparser is a contemporary dependency parsing machine learning-based system that shows a great accuracy. Nowadays the results are around 80 % (Labelled Attachment Score–LAS) for Spanish parsing and it seems that is not posible to beat these results. In this work we did a few studies about dependency parsers to find some techniques to improve this *de facto* limit of the accuracy. Firstly, we show some ideas and experiments based on the corpus size and/or sentences length, and finally we show the main idea of this work: the N-Version dependency parser, that is the idea to mix the action of some specific trained parsers, each parser is trained to achieve better accuracy in a specific task, and finally obtain better results in global dependency accuracy.

Palabras clave en Castellano

1. Procesamiento de Lenguaje Natural (PLN).
2. Aprendizaje Automático.
3. Análisis sintáctico de dependencias.
4. Entrenamiento basado en corpus.
5. Maltparser.
6. Análisis Sintáctico del Castellano.

Keywords in English

1. Natural Language Processing (NLP).
2. Machine learning.
3. Dependency parsing.
4. Corpus-based training.
5. Maltparser.
6. Spanish parsing.

Índice general

Capítulo 1

Introducción y objetivos

En este Capítulo se presenta una introducción del trabajo. Se describe en líneas generales lo que se aborda, y se define el análisis de dependencias y el entrenamiento basado en corpus. Proporcionando de esta manera, los conceptos generales para entender el resto del trabajo.

1.1. Descripción y objetivos del trabajo

En este trabajo se presenta un estudio elaborado para la mejora de la precisión del análisis de dependencias sintácticas.

Partiendo de unos resultados actuales muy buenos en el análisis de dependencias que se describen en el Capítulo 2, se pretenden mejorar, no sólo los resultados globales, sino obtener una precisión local más elevada. De esta manera el usuario final de un analizador de dependencias recibirá una solución más satisfactoria.

Para ello, se trata de buscar respuesta a las siguientes preguntas:

- ¿Qué factores del corpus de entrenamiento pueden afectar a la precisión en el análisis?
- ¿Es viable una mejora de la precisión mediante la manipulación del corpus de entrenamiento?
- ¿Es viable una mejora de la precisión mediante modificaciones del flujo de análisis?
- ¿Qué acciones para la mejora de la precisión son automatizables?

Se ha utilizado un generador de analizadores de dependencias automático, basado en aprendizaje automático mediante corpora de entrenamiento que contienen frases etiquetadas con análisis de dependencias.

Se han desarrollado una serie de técnicas de bajo y alto nivel, para conocer en profundidad el funcionamiento y los detalles de los analizadores de

dependencias, mediante las que, tras el proceso de investigación, se pudieron desarrollar algunas técnicas que promovieron una mejor calidad en el análisis de dependencias.

El material, tanto corpus utilizado como generador de analizadores de dependencias se describen en las Secciones 2.3.1 y 2.2.2, respectivamente y contenidos en el Capítulo 2. Y todos los experimentos y mejoras, se describen en el Capítulo 3.

En primer lugar, se estudiaron técnicas de alto nivel basadas en la homogeneidad de la precisión partiendo de un analizador no modificable. Para ello, se realizaron una serie de experimentos generando corpora de entrenamiento y análisis de diferentes tamaños y características, para luego realizar experimentos similares centrándose en las características de las frases contenidas en los corpora.

En segundo lugar, se estudió la viabilidad de un analizador N–Versiones, que consiste en la combinación de diferentes analizadores cada uno entrenado para analizar una parte de la frase diferente y obtener de esa manera un análisis de mejor calidad.

Finalmente, se muestra una aproximación a un algoritmo que envía las diferentes palabras de una frase al analizador más adecuado, convirtiendo dicho algoritmo en un analizador N–Versiones.

Las ideas que se plantean en este trabajo surgieron inspiradas por (McDonald y Nivre, 2007), combinando lo que se presenta en ese capítulo con el hecho de que una estrategia común en tolerancia a fallos es la implementación de sistemas N–Versiones. Habitualmente los sistemas N–Versiones realizan el mismo cómputo simultaneamente y luego se elige una de las n salidas. En este trabajo se presenta una alternativa, donde se elige a priori, cuales de los n sistemas van a computar cada parte y en cada instante, para posteriormente combinar las salidas de los mismos para obtener la salida final.

Por lo tanto, queda de manifiesto, la originalidad de la propuesta. Aunque existen trabajos en los que subyace la idea de la combinación de distintos sistemas, pero en el caso de etiquetadores de categorías gramaticales como (Sjöbergh, 2003) y (Gambäck et al., 2009). No así, en el caso de analizadores sintácticos de dependencias, donde el presente trabajo se puede considerar pionero en la materia.

A continuación se expone de manera general los dos conceptos fundamentales sobre los que versa este trabajo: el análisis sintáctico de dependencias y el aprendizaje automático basado en un corpus de entrenamiento.

1.2. Análisis Sintáctico de Dependencias

En esta sección se presenta el tema fundamental de estudio que se ha usado para el desarrollo de este trabajo: *El análisis de dependencias.* De

esta manera, se presenta una definición formal del concepto de dependencia, orígenes sobre gramáticas de dependencia y una serie de medidas de evaluación actuales que se usan para medir la precisión en el análisis de dependencias automático.

1.2.1. Descripción

La idea fundamental de *dependencia* está basada en que la estructura sintáctica de una frase consiste en relaciones binarias asimétricas entre las palabras de esa frase (Kübler, McDonald, y Nivre, 2009). Por tanto han de establecerse criterios para definir qué relaciones de dependencia existen, para distinguir de qué forma están relacionadas dos palabras en una frase y si esas relaciones están etiquetadas o no.

Por tanto, a partir de las relaciones asimétricas y los diferentes criterios. Dada una frase, en cualquier lenguaje, se puede establecer un árbol sintáctico de dependencias etiquetado como el del ejemplo que se muestra en la Figura 1.1. En este ejemplo se observa el análisis de dependencias de la frase en castellano: *"Le dijo a Auri que se iba al fútbol, ella le miró con los ojos fuera de las órbitas"*. Se puede razonar que dada esa frase y conociendo cierta información sintáctica de cada uno de los elementos que aparecen en la misma, se pueden establecer distintas relaciones de dependencia, donde la raíz de todas ellas es la acción principal de la frase.

Con la información contenida en un árbol de dependencias se pueden realizar múltiples tareas, como simplificación de textos (Caseli et al., 2009), reconocimiento de la implicación textual (Herrera, Peñas, y Verdejo, 2005), detección de conceptos negados en una frase (Ballesteros, Martín, y Agudo, 2010) o (Morante, Liekens, y Daelemans, 2008), etc. Pero eso no es lo que ocupa el desarrollo de este trabajo. Este trabajo se centra en la mejora de la precisión en el análisis de dependencias automático, basado en aprendizaje automático y orientado hacia un corpus concreto: AnCora (Mariona Taulé y Recasens, 2008), y un generador de analizador de dependencias basado en aprendizaje automático a partir de corpus concreto: Maltparser (Nivre et al., 2007).

1.2.2. Medidas de Evaluación

A la hora de medir la calidad en el análisis de dependencias, mediante un corpus de test, existen tres medidas de evaluación para indicar cuanta precisión hay entre el corpus de test manualmente etiquetado que se compara con el etiquetado automático de analizador. Para ello se han utilizado las siguientes 3 medidas, que son un estándar *de facto* en el análisis.

- **LAS: Labelled Attachment Score** (Nivre, Hall, y Nilsson, 2004), es el porcentaje de elementos presentes en el corpus de test, para los

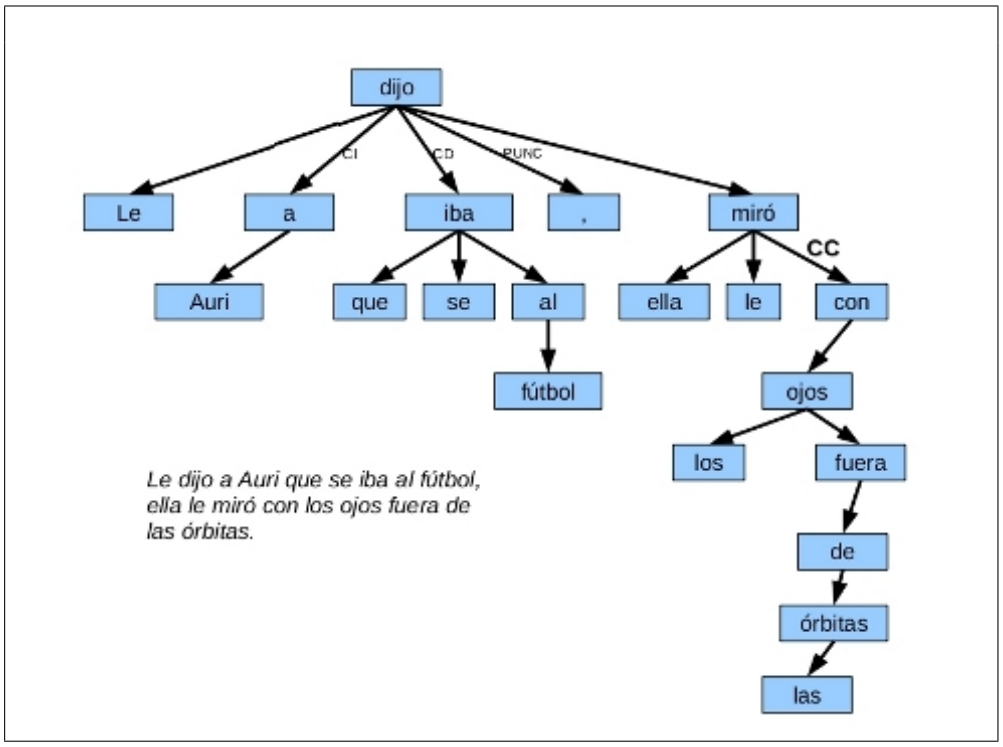

Figura 1.1: Ejemplo de una frase analizada según el paradigma de dependencias en formato de árbol de dependencias.

que el sistema ha predicho correctamente la etiqueta de dependencia y de que otro elemento depende.

- **UAS: Unlabelled Attachment Score** (Eisner, 1996), es el porcentaje de elementos presentes en el corpus de test, para los que el sistema ha predicho correctamente de que otro elemento depende.
- **LA: Label Accuracy** (Yamada y Matsumoto, 2003), es el porcentaje de elementos presentes en el corpus de test, para los que el sistema ha predicho correctamente la etiqueta de dependencia.

Por tanto si disponemos de un corpus de test que está compuesto de 4.991 palabras, o elementos, y después de analizarlo con nuestro analizador de dependencias hemos obtenido que hay 4.505 etiquetas bien puestas, 4.255 decisiones correctas donde se indica que una palabra depende de otra, y además sabemos que hay 4086 casos en los que se ha acertado en ambas cosas, los resultados serían los siguientes.

- LAS: (4086 / 4991) * 100 = 81,87 %
- UAS: (4255 / 4991) * 100 = 85,25 %

- LA: (4505 / 4991) * 100 = 90,26 %

Como se puede deducir de las definiciones de cada una de las medidas, la medida que tiene un carácter más general: Labelled Attachment Score (LAS) es, a la vez, la más importante, ya que engloba toda la evaluación posible en una sola medida. A lo largo de este trabajo, se muestran las tres medidas de evaluación pero en muchos casos se obviaran las medidas más secundarias y se ofrecerán resultados únicamente en LAS, ya que además esta medida más general, engloba los resultados de las otras dos.

1.2.3. Análisis Sintáctico de Constituyentes

El análisis de constituyentes es una alternativa diferente a la hora de analizar las características sintácticas de una frase, en este caso los elementos de la frase solo se encuentran en los nodos hoja del árbol, siendo el resto de nodos relaciones. El análisis de constituyentes es más natural para el tratamiento del análisis mediante gramáticas ya que se puede generar una gramática que reconozca los elementos presentes en una frase, donde el elemento principal será la frase, S como se puede ver en la Figura 1.2 y en el extracto de gramática (no está completa) de la misma figura.

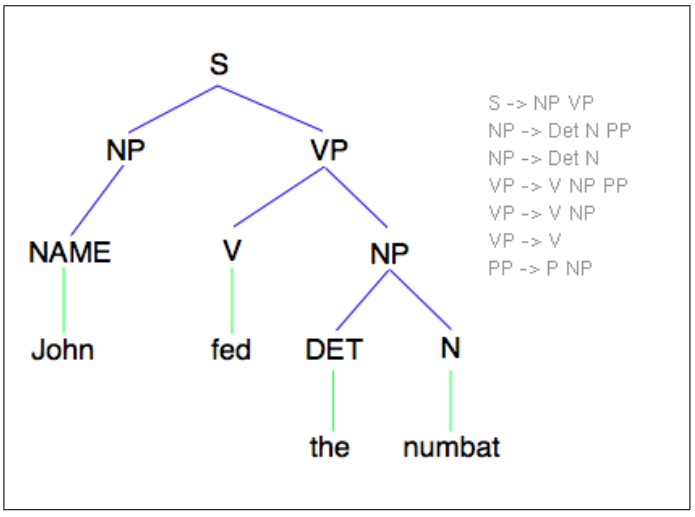

Figura 1.2: Frase analizada mediante análisis de constituyentes.

Como se ve en la figura, una frase tiene predicado y sujeto, que a su vez se dividen en otras ramas, hasta alcanzar los nodos hoja que son las propias palabras de la frase.

1.2.4. Análisis de Dependencias vs Análisis de Constituyentes

El análisis de constituyentes es un análisis sintáctico de oraciones que puede parecer más natural para un usuario que tenga conocimientos de

análisis sintáctico por eso es interesante exponer una breve comparativa entre ambos métodos para así poner de manifiesto un mayor entendimiento del concepto de análisis de dependencias. Es importante decir que existe un método automático (Gelbukh, Torres, y Calvo, 2005) para transformar un árbol de constituyentes en un árbol de dependencias. Por lo tanto, se puede afirmar que ámbos tipos de análisis son equivalentes.

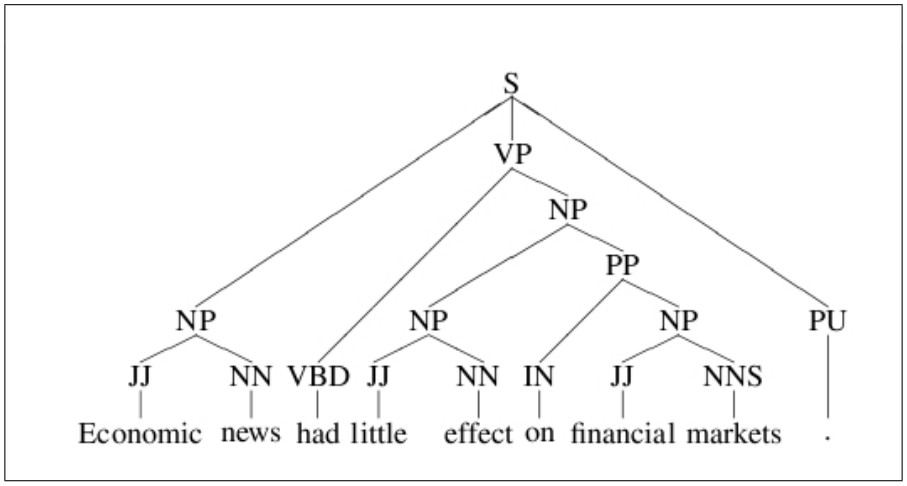

Figura 1.3: Frase analizada mediante análisis de constituyentes.

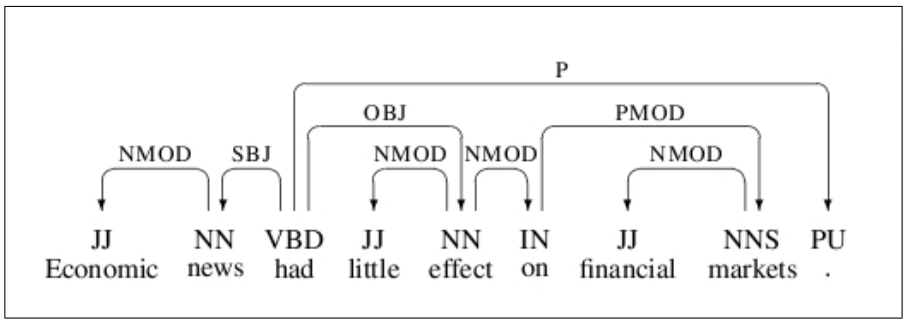

Figura 1.4: Frase analizada mediante análisis de dependencias.

Las figuras 1.3 y 1.4 muestran el mismo análisis, una en formato de análisis de constituyentes y la otra en formato de análisis de dependencias. En la Figura 1.4 se observa un árbol de dependencias en formato lineal. Se puede extender la información contenida en las flechas, dejando un nodo por palabra y el resultado final sería un árbol como el de la Figura 1.1 de la página 14.

La principal diferencia que se puede observar es que los árboles de constituyentes sólo contienen palabras en los nodos hoja del árbol, sin embargo, los árboles sintácticos de dependencias contienen palabras en cualquier parte

del árbol. Esta característica aporta mayor complejidad al análisis de dependencias y permite el análisis de lenguajes que tienen más libertad en el orden de las palabras, como el checo que aporta mucha libertad, o el castellano que es menos *libre*, pero no es tan restrictivo como el inglés.

Como se demuestra en (Kübler, Prokić, y Groningen, 2006) el análisis de ciertas lenguas es más factible mediante análisis de dependencias ya que tienen mayor libertad a la hora de colocar las palabras en una misma frase. Se observa que para el caso del alemán el análisis de dependencias permite relaciones con mayor distancia entre palabras y una mejor coordinación entre las mismas.

1.3. Aprendizaje Automático–Entrenamiento basado en corpus

En esta sección se hace un resumen del funcionamiento de los sistemas basados en aprendizaje automático y más concretamente aquellos que utilizan un corpus continente de una batería de ejemplos para aprender y ser capaz, después del aprendizaje de clasificar, etiquetar, etc. En este trabajo, se ha utilizado un sistema basado en aprendizaje automático, Maltparser que se describe en profundidad en la Sección 2.2.2 y un corpus de frases etiquetadas para el castellano, AnCora que se describe en la Sección 2.3.1.

El sistema basado en aprendizaje, dadas unas directrices de aprendizaje y un corpus de entrenamiento etiquetado, entrena analizadores específicos que son capaces de etiquetar y construir árboles de dependencia a partir de un corpus de test que tan sólo contiene información morfológica pero no información sintáctica.

Posteriormente, con el corpus de test ya analizado, se puede obtener la precisión en el análisis (las medidas de precisión se describen en la Sección 1.2.2) y de esta manera conocer, cómo se ha comportado el analizador de dependencias entrenado. Todo el proceso que se sigue, se puede observar en la Figura 1.5. Se expone, como el generador de analizadores de dependencias Maltparser recibe como entrada un modelo de características y un corpus de entrenamiento y devuelve un modelo entrenado capaz de analizar frases que se encuentra en el corpus de test. El modelo entrenado devuelve un corpus de test analizado y posteriormente, con un script de evaluación, se puede medir la calidad en el análisis.

1.4. Recapitulación

En este Capítulo 1 se han explicado de manera general los objetivos y se ha descrito a modo de introducción el trabajo que a continuación se expone. Para ello, además, se describen algunos conceptos generales que marcan las

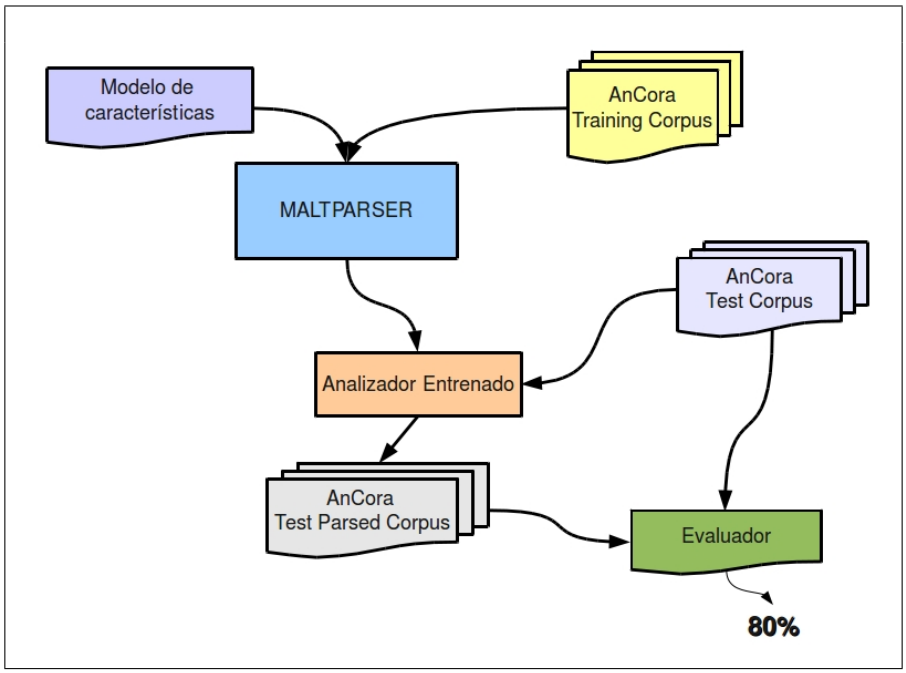

Figura 1.5: Proceso seguido para entrenar analizadores de dependencias y obtener resultados.

líneas base en el desarrollo de este trabajo, como es el Análisis de Dependencias, los sistemas basados en aprendizaje automático y el entrenamiento basado en un corpus de ejemplos.

Capítulo 2

Análisis Sintáctico de Dependencias

En este capítulo de la memoria se expone en qué estado se encuentra en la comunidad internacional el análisis sintáctico de dependencias automático, realizando en primer lugar un recorrido por los analizadores existentes más significativos, continuando con los corpora para el análisis de dependencias y finalmente sobre las CoNLL–X Shared Tasks, que son dos conferencias cuyo objetivo era ampliar el grado de avance en el análisis sintáctico de dependencias.

2.1. Introducción

Aproximadamente desde 2005 se ha experimentado en la comunidad dedicada al PLN (Procesamiento del Lenguaje Natural) un interés muy creciente en el análisis sintáctico de dependencias automático como herramienta auxiliar, muy probablemente por influencia de la eficiencia de Minipar (Lin, 1998), así como por la facilidad y conveniencia del procesamiento de las dependencias para su uso por programas de PLN. De este modo, en 2006 ya se estaba celebrando un evento internacional específico sobre análisis sintáctico de dependencias automático: la CoNLL X Shared Task on Multi-Lingual Dependency Parsing. Principalmente se organizó porque se desarrolló MaltParser (Nivre et al., 2007) y de algún modo querían ponerlo en comparación con otros trabajos. Antes de que se conociese que la Shared Task iba a tener lugar, pero sabiendo que existía MaltParser y algún analizador similar, fue cuando se inició el desarrollo de JBeaver (Herrera et al., 2007b) utilizando MaltParser como base. Tanto la Shared Task como JBeaver tenían común uno de sus objetivos: el desarrollo de analizadores sintácticos de dependencias para idiomas distintos del inglés (puesto que antes básicamente sólo existía Minipar). Actualmente existen para muchos idiomas distintos, como **MSTParser** (McDonald, Lerman, y Pereira, 2006), **WCDG Parser**(Foth,

Daum, y Menzel, 2004) o **CaBoCha** (Kudo y Matsumoto, 2002).

2.2. Analizadores Automáticos de Dependencias y Generadores de Analizadores de Dependencias

En esta sección se exponen los principales analizadores de dependencias disponibles en la actualidad, haciendo mención a Minipar ya que es un ejemplo significativo por ser precursor y por utilizar técnicas innovadoras y diferentes. Maltparser no es realmente un analizador de dependencias sino un generador de analizadores de dependencias basado en aprendizaje automático. Se hace especial mención a JBeaver, ya que una de sus funcionalidades es un analizador entrenado y libre para el análisis de dependencias del castellano.

2.2.1. Minipar como Ejemplo Significativo

Minipar (Lin, 1998) es un analizador de dependencias desarrollado por Dekang Lin. Como se ha dicho en la introducción, no es un analizador basado en aprendizaje automático y es exclusivo para el inglés. Aún así merece la pena estudiarlo, dada la enorme relevancia que ha tenido posteriormente, ya que aporta un buen resultado y una serie de ideas interesantes.

Minipar es un analizador de dependencias, al que se le proporciona un texto y genera el arbol de dependencias indicándonos para cada palabra la información necesaria para construir este arbol (identificador del nodo e identificador del nodo padre) así como alguna información adicional.

Con la aparición de los analizadores de amplia cobertura y, además, cada vez un número mayor de ellos, la evaluación de los analizadores es siempre muy importante. Para realizar la evaluación, se debe comparar los árboles etiquetados (answers) con los árboles marcados manualmente (keys). Lo que a lo largo del desarrollo de esta memoria entenderememos como corpus de entrenamiento y corpus de evaluación respectivamente. Como veremos, es objeto de debate cómo debe hacerse esta comparación. En Minipar se propone una evaluación basada en dependencias, donde las relaciones de dependencia, en lugar de otras alternativas serán las usadas en la comparación entre answers y keys.

Existe una versión web de Minipar[1] y existen distribuciones gratuitas para cualquier sistema operativo, aunque es recomendable su uso en un sistema libre como Linux ya que para este sistema operativo el código se encuentra compilado y para versiones bajo MS Windows requiere mayor esfuerzo. La versión web permite obtener el análisis de dependencias de una frase individual de manera simple y rápida.

[1] http://ai.stanford.edu

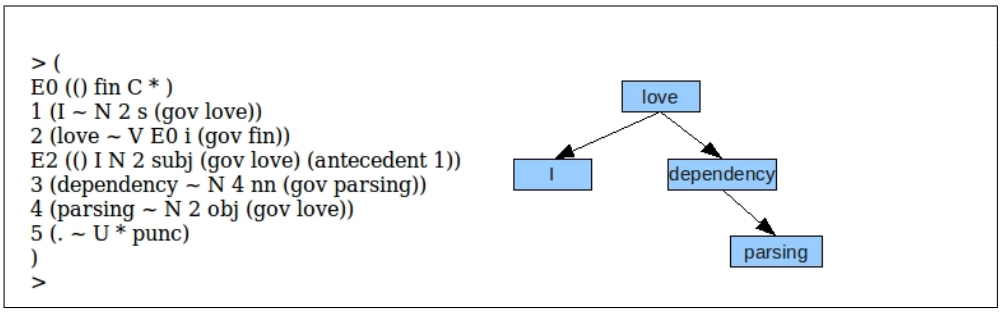

Figura 2.1: Una frase ejemplo analizada en inglés mediante la versión web de Minipar.

Como se puede observar en la Figura 2.1, la frase *"I love dependency parsing"* se analiza y se obtiene un árbol de dependencias completo y bien analizado (Minipar devuelve la parte codificada).

Los resultados de Minipar, a pesar de ser un analizador algorítmico no basado en aprendizaje, son bastante competitivos. Minipar cubre el 79 % de las relaciones de dependencia en el corpus SUSANNE (Sampson, 2002) con un 89 % de precisión para el análisis del inglés (Lin, 1998).

2.2.2. Maltparser

MaltParser (Nivre et al., 2007) es un sistema de análisis sintáctico de dependencias que pretende ser independiente del idioma. Está dirigido por el profesor Joakim Nivre de la Universidad de Växjö, en Suecia . Realmente es un generador de analizadores de dependencias y no un analizador en sí.
Su creación estuvo motivada porque los analizadores sintácticos solían estar demasiado ajustados a un cierto idioma y como consecuencia funcionan mal para otros. También busca una solución para el indeterminismo de los analizadores de dependencias.
Maltparser, a diferencia de Minipar, está basado en aprendizaje. Además es aplicable a cualquier idioma; por tanto podemos crear tantos analizadores como entrenamientos hagamos. Maltparser genera modelos entrenados para el corpus que le indiquemos pudiendo utilizarlo posteriormente.
El grupo de Nivre y su Maltparser se han convertido en una referencia internacional en análisis de dependencias, estando presentes en las reuniones y conferencias más importantes como la CoNLL–X Shared Task; podemos tomar los datos obtenidos en la CoNLL–X Shared Task por el grupo de la Universidad de Växjö de Suecia (aparte de otros grupos) que obtuvo los siguientes resultados que se muestran en la Tabla 2.1, en los idiomas que se usaron para la evaluación:

Como se puede observar en la Tabla 2.1 los resultados globales obtenidos por Maltparser son muy destacables, por ejemplo los resultados obtenidos tras el análisis del checo que tiene una mayor dificultad que otras lenguas

ya que tiene más libertad en el orden de las palabras.

Tabla 2.1: Resultados del Sistema Maltparser en la CoNLL–X Shared Task

Idioma	**Resultados**
Alemán	85,82 %
Búlgaro	87,41 %
Checo	78,42 %
Chino	86,92 %
Danés	84,77 %
Esloveno	70,30 %
Español	**81,29 %**
Holandés	78,59 %
Japonés	91,65 %
Portugués	87,60 %
Sueco	84,58 %
Turco	65,68 %
Media	80,74 %

Funcionamiento de Maltparser

En esta subsección se exponen en primer lugar los pasos a seguir para entrenar un modelo y poder analizar un corpus de test mediante Maltparser. En primer lugar, se necesita un sistema operativo tipo Unix que tenga librerías estándar de C instaladas. El acceso simple a Maltparser, aunque existe alguna versión con interfaz gráfica, se hace desde el terminal.

Maltparser debe tener 2 archivos de opciones, uno para entrenamiento y otro para análisis, donde se indica qué corpus se utiliza, algoritmo, archivo de salida, versión del algoritmo, etc.

Para ejecutar Maltparser en modo entrenamiento, haciendo que Maltparser genere un modelo entrenado (que consiste en un conjunto de archivos que modifican la configuración por defecto, ajustados a los parámetros del corpus y del módelo de características), se ejecuta un comando mediante la consola Unix y de esa manera obtenemos el resultado deseado, o bien el analizador entrenado, o bien el análisis generado.

Algoritmo de Nivre

El algoritmo implementado en Maltparser es una versión para análisis de dependencias de un analizador "shift-reduce". Es similar a algoritmos para gramáticas libres de contexto, es decir, se utiliza una pila como estructura de datos y se procede como sigue: el elemento más a la izquierda se apila en

la pila, que se usa a lo largo de todo el proceso. También hay una reducción que es basicamente sacar un elemento de la pila. Además hay dos formas de introducir relaciones de dependencia etiquetadas. Por un lado puede ser: la dependencia está en la cabeza de la pila y el elemento del que depende es el siguiente token de entrada. O bien puede ser: la dependencia es el siguiente token de entrada y el elemento del que depende está en la cabeza de la pila. Otros grupos, aparte del grupo de Nivre, usaron su algoritmo en la CoNLL X Shared Task, al menos una versión básica del mismo, como pueden ser el grupo de Johansson y Nugues (Johansson y Nugues, 2006) o el grupo de Yu-Chieh Wu, Wu, Yue-Shi Lee y Jie-Chi Yang(Wu, Lee, y Yang, 2006).

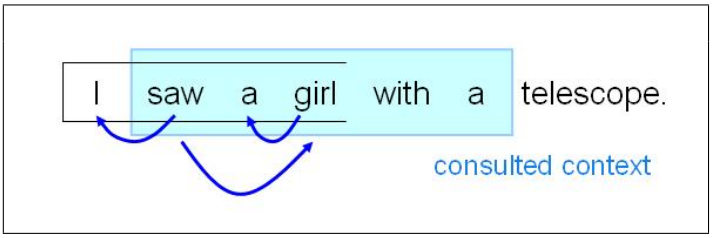

Figura 2.2: Formato de decisiones del método de Nivre.

El algoritmo de Nivre tiene un coste lineal O(n) sobre el número de nodos, ya que recorre cada nodo tan sólo una vez recorriendo el árbol en profundidad. Pero quizá esto puede ser un punto negativo de este algoritmo ya que no puede rehacer el trabajo ni revisarlo, es decir, no aplica técnicas de vuelta atrás sino que recorre el árbol completo tomando las decisiones rapidamente, y una sóla vez por elemento, según lo aprendido. En la Figura 2.2 se observa el formato de decisiones del método de Nivre, donde se expone precisamente lo mencionado anteriormente, las decisiones en el algoritmo se toman una sóla vez para cada palabra y sin contar con una etapa de revisión.

Es sin duda, una aproximación muy interesante y, conocidos los resultados que obtiene, se demuestra que no es necesario realizar cálculos excesivamente complejos. Cabe decir que analizar un corpus de considerable tamaño lleva mucho tiempo, por tanto es interesante tener aproximaciones capaces de simplificar el proceso.

Modos de entrenamiento

Maltparser utiliza modelos de características basados en historia (*history-based feature models*[2]) para predecir la siguiente acción a realizar en la derivación determinista para generar el árbol de dependencias, lo que significa que Maltparser utiliza características del árbol de dependencias que se está construyendo, y además usa características del etiquetado previo necesario para construirlo. Por tanto, las características se definen en términos

[2]http://w3.msi.vxu.se/nivre/research/MaltParser.html

del lexema (LEX), categoría gramatical (POS) o relación de dependencia (DEP) entre ese elemento y su cabecera, entre otros.

Para ello se usan tres estructuras de datos diferentes: una pila (STACK), la cadena de entrada (INPUT) y el contexto (CONTEXT), y una serie de funciones auxiliares para definir de qué modo debe generarse: HEAD (cabecera), LC (elemento más a la izquierda que depende del nodo que se está tratando), RC (elemento más a la derecha que depende del nodo que se está tratando), RS (desplazamiento derecha), LS (desplazamiento izquierda). Un modelo de características se define en un archivo y en el archivo de opciones debe indicarse a Maltparser que modelo concreto es el que debe seguir para realizar el aprendizaje y construir el producto final: el analizador de dependencias.

El archivo que contiene el modelo es realmente una matriz que contiene un número indefinido previamente de filas y un número fijo de columnas, 7. A continuación se expone el significado de cada una de las columnas y se muestra un ejemplo sencillo para su comprensión, en la la Figura 2.3.

```
POS     STACK   0       0       0       0       0
POS     INPUT   1       0       0       0       0
POS     INPUT   0       -1      0       0       0
DEP     STACK   0       0       1       0       0
DEP     STACK   0       0       0       -1      0
```

Figura 2.3: Ejemplo sencillo de un modelo de características almacenado en un fichero con extensión .par.

- La primera columna indica sobre qué etiqueta del corpus debe fijarse el sistema.
- La segunda columna indica qué estructura de datos se utiliza.
- La tercera columna define el desplazamiento i, que sólo puede ser positivo e identifica la posición de la estructura de datos identificada en la segunda columna (STACK[i], INPUT[i] o CONTEXT[i]).
- La cuarta columna define un desplazamiento lineal i, que puede ser positivo o negativo y se refiere a las posiciones de los elementos en la cadena de entrada.
- La quinta columna define un desplazamiento i en términos de la cabecera del token concreto, debe ser positivo e identifica i aplicaciones de la función de cabecera (HEAD function) al elemento identificado por los desplazamientos anteriores.
- La sexta columna define el número de veces, i, que deben aplicarse las funciones LC o RC. Se toma el valor absoluto.

- La séptima columna define el número de aplicaciones, i, que debe aplicarse las funciones LS o RS. Se toma el valor absoluto.

Una tarea importante a la hora de desarrollar los experimentos realizados en esta Tesis de Fin de Máster, que se describen en el Capítulo 3, ha sido la búsqueda de diferentes modelos de características para los distintos analizadores de dependencias que se han entrenado y usado para analizar corpora de test. Aún así el modelo de entrenamiento que más se ha utilizado es el propuesto por (Nivre et al., 2006), en la CoNLL–X Shared Task, y que se muestra en la Figura 2.4. El grupo de (Nivre et al., 2006) participó con un modelo para cada uno de los lenguajes propuestos en la CoNLL–X Shared Task y se puede observar la complejidad del modelo propuesto para el castellano.

```
POS     STACK
POS     INPUT
POS     INPUT   1
POS     INPUT   2
POS     INPUT   3
POS     STACK   1
POS     STACK   0       0       1
POS     STACK   0       0       0       -1
POS     STACK   0       0       0       1
POS     INPUT   0       0       0       -1
POS     STACK   0       1
POS     INPUT   0       -1
POS     STACK   2
FEATS   STACK
FEATS   INPUT
FEATS   INPUT   1
FEATS   STACK   0       0       1
DEP     STACK
DEP     STACK   0       0       0       -1
DEP     STACK   0       0       0       1
DEP     INPUT   0       0       0       -1
LEX     STACK
LEX     INPUT
LEMMA   STACK
LEMMA   INPUT
LEMMA   INPUT   0       0       0       -1
CPOS    STACK
CPOS    INPUT
CPOS    INPUT   1
CPOS    STACK   0       0       1
```

Figura 2.4: Modelo propuesto para el análisis del castellano, contenido en el fichero spa.par

Además de las instrucciones y modelos específicos para algunas lenguas, existen una serie de modelos básicos que suelen comportarse bien con la

mayoría de los corpora. Estos modelos se llaman m2.par, m3.par y m4.par y pueden encontrarse junto con la documentación de Maltparser[3]

Justificación del uso de Maltparser

Maltparser ha sido elegido en nuestro trabajo para realizar los experimentos con el fin de generar los distintos analizadores de dependencias para poder medir la calidad del análisis. Es decir, se ha utilizado como herramienta principal en los desarrollos y experimentos que se exponen en el Capítulo 3.

Maltparser aporta soluciones de gran calidad a la hora de entrenar analizadores de dependencias y los resultados obtenidos por analizadores globales, entrenados por Maltparser, son muy buenos. Maltparser ha sido utilizado no sólo por grupos de investigación relacionado con los desarrolladores de Maltparser sino también por otros. En las dos Shared Task donde se compitió para obtener el mejor análisis de dependencias, el grupo de Nivre (desarrolladores de Maltparser) estuvo siempre en las primeras posiciones para cada lengua. Por tanto, en definitiva, se puede afirmar, que Maltparser es de los mejores sistemas basados en aprendizaje para generar analizadores de dependencias. Su carácter libre y las numerosas opciones que aporta como modos de aprendizaje han sido otros factores importantes a tener en cuenta.

2.2.3. JBeaver

JBeaver (Herrera et al., 2007b) es un sistema de análisis y generación de analizadores de dependencias, realizado como proyecto de fin de carrera por alumnos de la Universidad Complutense en el curso académico 2006/07, y dirigido por miembros del grupo NIL[4] de la misma universidad, alguno de los cuales ha colaborado en la dirección de este mismo proyecto de investigación. En el momento del inicio del desarrollo de JBeaver, la existencia de Minipar y MaltParser suponían un incentivo importante para querer desarrollar un analizador de dependencias para el castellano. Gracias a MaltParser era viable la realización de un generador de analizadores de dependencias para el castellano, con pocos recursos.

JBeaver es un sistema programado en Java, lo que lo convierte en un sistema portable, además de autónomo, y fácil de usar. Con JBeaver podemos crear corpora de entrenamiento, entrenar un sistema automático de aprendizaje y por último realizar análisis y evaluarlos tanto gráfica como estadísticamente, lo cual aporta una potencia y visibilidad muy elevadas.

JBeaver incluye un modelo entrenado por defecto, mediante un corpus de entrenamiento obtenido a partir de Cast3LB (Navarro et al., 2003), lo que

[3] http://w3.msi.vxu.se/nivre/research/MaltParser.html
[4] Natural Interaction Language–http://nil.fdi.ucm.es

lo convierte en un analizador posible de utilizar sin necesidad de disponer de corpora de entrenamiento.

JBeaver se divide en tres módulos funcionales:

Módulo de entrenamiento .

JBeaver, en su versión inicial, utiliza Maltparser 0.4. Este módulo se encarga del entrenamiento de la herramienta Maltparser y de proporcionar toda la configuración necesaria para ello. En el momento del inicio de este proyecto JBeaver contaba con un corpus de entrenamiento basado en análisis de constituyentes (Cast3LB), el cual se transforma mediante el algoritmo de Gelbukh (Gelbukh, Torres, y Calvo, 2005) a un corpus basado en análisis de dependencias. Lo cual permite el entrenamiento de la herramienta Maltparser 0.4. Como se ha dicho en el primer capítulo de esta memoria, se desea entrenar JBeaver con un corpus mejorado con respecto a los actualmente disponibles para obtener una mayor calidad en el análisis generado.

Módulo de análisis .

Este módulo realiza las siguientes funciones: etiqueta cada token de un texto con sus *part-of-speech* (categorías gramaticales), analiza el texto de entrada, ya etiquetado, gracias al modelo creado en el entrenamiento, y, por último, evalúa la calidad del análisis, usando las métricas: Label Attachment score (LAS), Unlabel Attachment score (UAS) y Label Accuracy (LA).

Módulo gráfico .

JBeaver proporciona la posibilidad de mostrar los árboles de dependencias de forma gráfica y legible para el usuario. Lo que convierte este módulo en un complemento del anterior. En la Figura 2.5 se ob-

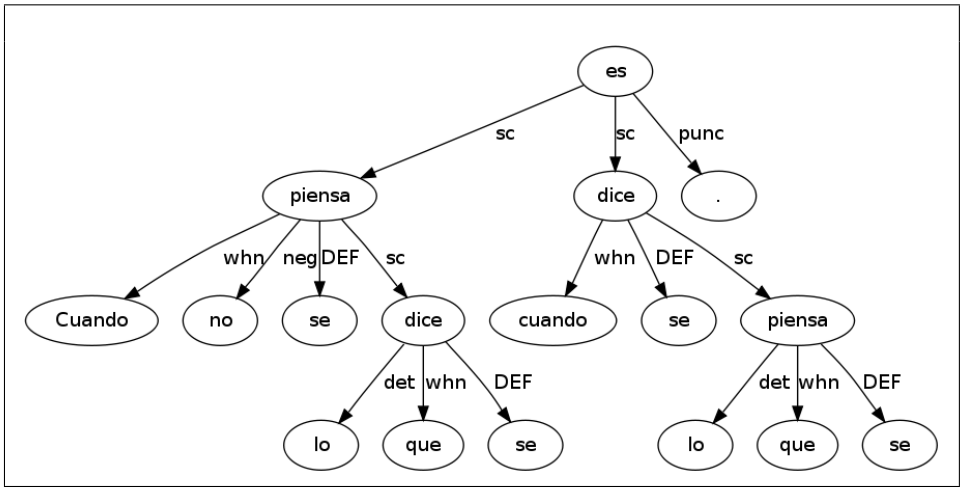

Figura 2.5: Salida del módulo gráfico de JBeaver.

serva la salida del módulo gráfico para un árbol de dependencias de la frase: "*Cuando no se piensa lo que se dice es cuando se dice lo que se piensa*".

2.3. Corpora para el Análisis Automático de Dependencias

En esta sección se expone los corpora disponibles para el entrenamiento y análisis de analizadores de dependencias. Se describe en primer lugar el corpus AnCora, que está disponible en castellano y catalán y es actualmente un completo corpus orientado a sistemas basados en aprendizaje automático. A continuación se exponen diferentes corpora existentes para el análisis de dependencias para algunos idiomas, pero en este caso de manera más breve.

Actualmente, los corpora para el análisis automático de dependencias se pueden agrupar en 2 grandes grupos. Corpora que son de origen basados en análisis de dependencias, o corpora que en sus inicios estaba basado en otro tipo de formato, como análisis de constituyentes, y que se ha transformado de manera automática (algunos incorporan una revisión manual posterior). AnCora pertenece al segundo grupo, por lo que puede ser una interesante sugerencia de trabajo futuro el obtener un corpus genuinamente construido para análisis de dependencias en castellano.

Algunos corpora existentes actualmente y que fueron etiquetados en su origen con análisis de dependencias fueron:

- **Prague Dependency Treebank** (Böhmová et al., 2001). Este corpus está etiquetado en 3 niveles: morfológico, sintáctico y gramatical.
- **Prague Arabic Dependency Treebank** (Hajič y Zemánek, 2004). Consiste en un corpus anotado con análisis de dependencias para el árabe moderno.
- **Danish Dependency Treebank** (Kromann, 2003). El etiquetado de este corpus está basado en Gramáticas Discontinuas (Buch-Kromann, 2005).
- **Floresta Sintá(c)tica** (Afonso et al., 2002). Corpus anotado para el portugués.

2.3.1. AnCora

AnCora (Mariona Taulé y Recasens, 2008) es un corpus, que cuenta con alrededor de 500.000 palabras en español (también disponible en Catalán). Está construido incrementalmente desde el Corpus 3LB. El corpus se encuentra etiquetado automáticamente y el análisis de dependencias que muestra

se ha comprobado manualmente.

AnCora es el resultado de enriquecer el corpus 3LB llevándolo hasta 500.000 palabras y añadiéndole información semántica en diferentes niveles: estructuras argumentales, roles temáticos, clases semánticas, entidades nombradas y otro tipo de características sintácticas y semánticas.

AnCora está en formato CoNLL[5] (en la sección 2.4.1 se explica con mayor detalle en que consiste) y en la Figura 2.6 se observa como ejemplo la frase: *Le dijo a Auri que se iba al fútbol, ella le miro con los ojos fuera de las órbitas* en formato CoNLL presente en el corpus AnCora.

```
1       Le      él      p       pp      num=s|per=3|gen=c|case=d        2
2       dijo    decir   v       vm      num=s|per=3|mod=i|tmp=s 0       ROOT
3       a       a       s       sp      for=s   2       CI      2       CI
4       Auri    Auri    n       np      _       3       _       3       _
5       que     que     c       cs      _       7       _       7       _
6       se      él      p       p0      per=3   7       _       7       _
7       iba     ir      v       vm      num=s|per=3|mod=i|tmp=i 2       CD
8       al      al      s       sp      gen=m|num=s|for=c       7       CC
9       fútbol  fútbol  n       nc      gen=m|num=s     8       _       8
10      ,       ,       F       Fc      _       2       PUNC    2       PUNC
11      ella    él      p       pp      num=s|gen=f|per=3       13      SUJ
12      le      le      p       pp      num=s|per=3|gen=c|case=a        13
13      miró    mirar   v       vm      num=s|per=3|mod=i|tmp=s 2       _
14      con     con     s       sp      for=s   13      CC      13      CC
15      Los     Los     d       da      gen=m|num=p     16      _       16
16      ojos    ojo     n       nc      gen=m|num=p     14      _       14
17      fuera   fuera   r       rg      _       16      _       16      _
18      de      de      s       sp      for=s   17      _       17      _
19      las     el      d       da      gen=f|num=p     20      _       20
20      órbitas órbita  n       nc      gen=f|num=p     18      _       18
```

Figura 2.6: Una frase etiquetada del corpus AnCora.

Metodología para el desarrollo de AnCora

Los desarrolladores de AnCora, realizaron un proceso complejo para obtener el resultado final. Todas las etapas en el proceso de desarrollo tuvieron una etapa de verificación manual, mediante un grupo de lingüistas lo que asegura la calidad del etiquetado final. Cabe destacar que la base de AnCora era un corpus anotado con análisis de constituyentes y la mayor labor fue la programación de un algoritmo de paso de análisis de constituyentes a análisis de dependencias. Cabe destacar que los desarrolladores de JBeaver generaron un corpus de dependencias (Herrera et al., 2007a) partiendo del mismo corpus de constituyentes que los desarrolladores de AnCora utilizaron, utilizando para ello el algoritmo de Gelbukh (Gelbukh, Torres, y Calvo, 2005).

[5]http://nextens.uvt.nl/ conll/

Formatos de etiquetado en AnCora

Para indicar qué tipo de relaciones de dependencia existen entre dos palabras cualesquiera en una frase, se hace uso de las etiquetas de dependencia. Por ejemplo, la etiqueta Complemento Directo de un verbo. AnCora por tanto, dispone de un conjunto de etiquetas de dependencia, donde la frecuencia de las mismas no es igual, ya que hay ciertas relaciones de dependencia que son muy importantes y otras cuyo papel es meramente circunstancial.

Como se puede ver en la Tabla 2.2, donde se muestra cada etiqueta con su nombre, hay muchas relaciones de dependencia diferentes que aportan gran cantidad de opciones a la hora de analizar una frase mediante análisis sintáctico de dependencias.

Tabla 2.2: Etiquetas de AnCora.

Etiqueta	**Función**
ATR	Atributo
CAG	Complemento Agente
CC	Complemento Circunstancial
CD	Complemento Directo
CD.Q	Complemento Directo Cuantitativo
CI	Complemento Indirecto
CPRED.SUJ	Complemento Predicativo Sujeto
CPRED.CD	Complemento directo predicativo
CREG	Complemento Preposicional
ET	Elemento textual
IMPERS	Marca impersonal
MOD	Modificador de la acción
NEG	Elemento negativo
PASS	Marca pasiva
PUNC	Signo de puntuación
ROOT	Cabecera de la frase
SUJ	Sujeto
VOC	Vocativo

A continuación se muestra un ejemplo para cada una de las etiquetas, para que de esa manera se entienda mejor el comportamiento de cada una, y se aprecie la riqueza expresiva con la que se cuenta a la hora de diseñar un árbol de dependencias sintácticas.

ATR: *Estamos **preparados** y no tenemos miedo. Preparados* es el atributo (ATR) de *Estamos.*

CAG: *Los retos fijados **por** el Gobierno. Por* y todo lo que depende de *Por*: *(el Gobierno)* es el Complemento Agente de *Los retos fijados.*

CC: *Trasladó el material **a** Madrid.* La preposición *a* y todo lo que depende de la misma, en este caso *Madrid*, son el complemento circunstancial (CC) de la acción principal *Trasladó.*

CD: *No esperó su **respuesta**. Respuesta* es el Complemento Directo de la acción principal, *esperó.*

CD.Q: *El Banco tenía una **plantilla de 14000 empleados**.* La *plantilla* es el Complemento Directo Cuantitativo de El *Banco.*

CI: *Una de las llamadas fue dirigida **al entrenador del equipo**. El entrenador* es el Complemento Indirecto de *Una de las llamadas.*

CPRED.SUJ: *Se considera **importante** que el euro no vuelva a bajar. Importante* es el Complemento Predicativo Sujeto de la acción principal: *(considera).*

CPRED.CD: *Se considera **anticonstitucional**. Anticonstitucional* es el Complemento Directo Predicativo de la acción principal: *considera.*

CREG: *El senado someterá **a** votación a primeros de este mes.* La preposición *a* y todo lo que depende de la misma (*primeros de este mes*) es el Complemento Preposicional de la acción principal.

ET: ***Pero** Germán es un hombre apasionado. Pero* es el Elemento Textual que depende de la acción principal: *es.*

IMPERS: ***Se** trata de un hombre sin escrúpulos. Se* es la marca impersonal y depende de la acción principal: *trata.*

MOD: *No hablaríamos **ya** de la renta per cápita. Ya* modifica la acción principal.

NEG: ***No** esperó su respuesta. No* niega la acción principal: *esperó.*

PASS:*La verdadera heroicidad **se** construye calladamente. Se* es una marca pasiva.

PUNC: Cualquier signo de puntuación como puntos, comas, etc.

ROOT: La acción principal de toda frase, en su defecto, la primera palabra. En la frase *María habla Inglés*, el verbo *habla* es la acción principal de la frase.

SUJ: *Como hizo **María**. María* es el sujeto de la acción principal: *hizo.*

VOC: ***Señor** Caballero, dedíquese a lo suyo. Señor* es el vocativo de *Caballero.*

Justificación del uso de AnCora

AnCora es un corpus muy extenso cuyo resultado final ha sido manualmente corregido por un grupo de lingüistas, lo que aporta seguridad en la calidad del análisis. Las frases son de diferentes dominios, lo que no convierte al trabajo en el análisis de un dominio propio sino general. Además se ha utilizado en numerosas ocasiones como corpus para entrenamiento y evaluación de analizadores de dependencias y existe una versión en un formato que Maltparser, el generador de analizadores de dependencias que se usó para la realización de este proyecto, acepta que se ha descrito al inicio de esta sección.

2.4. CoNLL Shared Tasks

Los organizadores de la Conferencia de Lenguaje Natural basada en Aprendizaje Automático (Conference on Computational Natural Language Learning: CoNLL) seleccionan una tarea cada año y los participantes compiten y prueban sus diferentes sistemas de aprendizaje usando los mismos corpora de entrenamiento y análisis. De esta manera, puede establecerse el grado de avance con datos reales y experimentales. Además de generar una serie de repositorios continentes de sistemas, corpora y diferentes elementos generados por los grupos presentes.

En 2006 y 2007, se decidió tratar el tema del análisis automático de dependencias. En 2006, se centró en el análisis multilingüe y en 2007 se crearon dos tareas diferentes, una fue análisis multilingüe, de nuevo y otra la adaptación al dominio. Para este trabajo interesa en mayor medida la conferencia de 2006, ya que se utilizó el castellano como uno de los idiomas que se analizaron y en la misma se presentó el corpus AnCora y el sistema Maltparser 0.4. Ambos productos se han utilizado para el desarrollo de los experimentos que se describen en este trabajo.

A continuación se describen las dos conferencias, 2006 y 2007, centrando los esfuerzos en la reunión de 2006.

2.4.1. CoNLL–X Shared Task (2006)

La CoNLL X Shared Task es el décimo de una serie de eventos organizados por SIGNLL, que cada año se selecciona una temática sobre la que presentar información lo que convirtió a la CoNLL X Shared Task en un evento internacional específico sobre análisis de dependencias, que se celebró en la ciudad de Nueva York, en junio de 2006. Tenía como objetivo principal establecer, y definir, el estado del arte sobre análisis de dependencias. Se presentaron un total de 52 artículos de los cuales solo 18 se aceptaron, lo que señala la elevada exigencia del evento.

La CoNLL X Shared Task solicitaba que los analizadores fueran facilmente parametrizables, esperaba que fueran multilingües, proporcionaba un marco de evaluación para poder evaluar los analizadores en diferentes lenguas, así como un formato de archivos de entrada para los analizadores. Su objetivo principal fue medir la calidad en el análisis sintáctico de dependencias.

Cada año la Conferencia de Aprendizaje Automático sobre Lenguaje Natural (CoNLL) versa sobre un tema, la décima CoNLL Shared Task el tema fue el análisis de dependencias multilingë (Buchholz y Marsi, 2006). El objetivo de esta reunión fue aumentar el grado de avance disponible en análisis de dependencias. La CoNLL Shared Task propuso un marco de evaluación de medida para evaluar los distintos analizadores presentados capaces de analizar 13 lenguajes diferentes, incluido el castellano. Los sistemas se puntuaban computando su capacidad de etiquetar correctamente las dependencias (Label Attachment Score) LAS , es decir, que el sistema haya predicho correctamente tanto de que palabra depende como la relación de dependencia con ella.(Nivre, Hall, y Nilsson, 2004). También se tuvieron en cuenta UAS(Unlabelled Attachment Score), donde sólo se toma en consideración la decisión de que palabre depende otra palabra (Eisner, 1996) y, finalmente LA(Label Accuracy) que es el porcentaje de relaciones de dependencias bien etiquetadas (Yamada y Matsumoto, 2003).

El lenguaje sobre el que se investiga y trata en este trabajo es el Castellano. Para este lenguaje los resultados obtenidos por los 19 participantes variaron entre un 47.0 % y un 82.3 % LAS, con un valor medio de 73.5 %. El corpus de entrenamiento utilizado para el español fue AnCora (Marionа Taulé y Recasens, 2008). Los resultados de la evaluación para el castellano se encontraban en la media. Los dos grupos participantes con un mejor resultado para el castellano fueron: el grupo de McDonald (McDonald, Lerman, y Pereira, 2006) (82.3 % LAS) y el grupo de Nivre (Nivre et al., 2006) (81.3 % LAS). En este trabajo centramos gran parte de la investigación en trabajos realizados por el grupo de Nivre, empezando por el sistema de entrenamiento que usamos para realizar los experimentos (Maltparser). Otros participantes que usaron el algoritmo de Nivre en la CoNLL–X Shared Task fueron el grupo de Johansson (Johansson y Nugues, 2006) y el grupo de Wu (Wu, Lee, y Yang, 2006). Sus resultados para el castellano fueron 78.2 % (séptima plaza) y 73.2 % (decimotercera plaza), respectivamente. La evaluación muestra que la aproximación dada por Nivre aporta un análisis competitivo para los lenguajes estudiados.

Aproximaciones destacadas antes de la CoNLL–X Shared Task

Collins y colaboradores (Collins, 1996) .

El analizador de Collins funciona con corpora etiquetados mediante el paradigma de estructura de frase. Para entrenar este analizador para el checo,que tiene un orden libre de las palabras, Collins et al. transfor-

maron la estructura de dependencias en una estructura de frase creando no terminales y etiquetándolos basándose en la categoría gramatical de la cabecera. Por ejemplo, si la cabeza es un nombre, la etiqueta del no terminal será NP, si la cabeza es un adjetivo, la etiqueta será AP.

Eisner (Eisner, 1996) .

Los modelos presentado por Eisner en 1996 sienta las bases de muchas de las aproximaciones presentadas en la CoNLL–X Shared Task. Eisner presentó tres modelos basados en algoritmos probabilistas cuyo coste era cúbico O(n3). Estos modelos estaban basados en algoritmos de dependencias que no etiquetaban las aristas, aunque posteriores aproximaciones incluyen el etiquetado de aristas, es decir, las dependencias.

El primer modelo presentado por Eisner se basa en la afinidad léxica entre los elementos de una frase según las palabras modificadoras de una frase. El segundo modelo consiste en colocar las palabras aleatoriamente para posteriormente seleccionar de forma probabilística la situación más adecuada. El tercer modelo de Eisner es similar al modelo usado por Collins; es un modelo de generación recursiva dando prioridad a la estructura sintáctica de la frase. Este modelo fue el que obtuvo mejores resultados de los tres empleando un corpus en inglés.

Diferentes aproximaciones presentadas a la CoNLL–X Shared Task

En este apartado se exponen las distintas aproximaciones que se presentaron a la CoNLL X Shared Task, describiendo los algoritmos implementados de forma breve. En el Apéndice A.2, se observan los resultados generales para cada uno de los grupos presentes en la CoNLL X Shared Task.

Nivre y otros (Nivre et al., 2006) .

El algoritmo es una versión para análisis de dependencias de un analizador "shift-reduce". Es similar a algoritmos para gramáticas libres de contexto; el elemento más a la izquierda se apila en una pila, que se usa a lo largo de todo el proceso. También hay una reducción que es basicamente sacar un elemento de la pila. Hay dos formas de introducir relaciones de dependencia etiquetadas. Por un lado puede ser: la dependencia está en la cabeza de la pila y el elemento del que depende es el siguiente token de entrada. O bien puede ser: la dependencia es el siguiente token de entrada y el elemento del que depende está en la cabeza de la pila.
El algoritmo que se describe aquí está implementado en MaltParser.

Matsumoto y otros (Cheng, Asahara, y Matsumoto, 2006) .

En esta aproximación las estructuras de dependencias se construyen

recorriendo la frase de izquierda a derecha, decidiendo repetidamente para cada par de palabras adyacentes cuál de esas palabras depende de la otra; si la comprobación tiene éxito entonces se eliminan de la entrada. El algoritmo original produce dependencias sin etiquetar, pero podría ser facilmente modificado prediciendo qué etiqueta le corresponde, sabiendo la decisión que se tomó sobre su dependencia. En la Figura 2.7 se observa el método de decisiones del algoritmo de Matsumoto, que incluye una etapa de revisión.

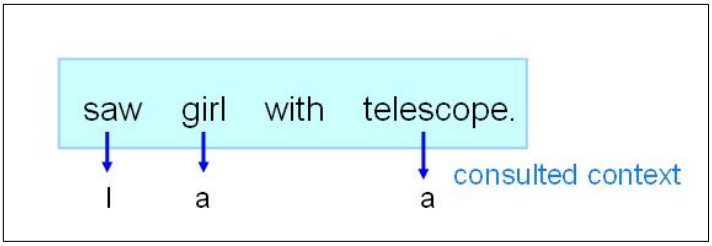

Figura 2.7: Formato de decisiones del método de Matsumoto.

El algoritmo utilizado realiza un recorrido primero en anchura por lo que su coste es cuadrático O(n2) en el peor de los casos, aunque eso le asegura una elevada precisión.

McDonald y otros (McDonald, Lerman, y Pereira, 2006) .

En el algoritmo de McDonald y colaboradores[6] (McDonald, Lerman, y Pereira, 2006), cuyos resultados en la CoNLL–X Shared Task fueron muy destacables, para cada par de palabras X e Y en la frase se comprueba como de probable es que X dependa de Y. Se trata de un algoritmo de programación dinámica y se selecciona el etiquetado más probable de los posibles. Por tanto es un algoritmo probabilista Para cada palabra se asigna una función etiquetadora que asigna en cada paso la etiqueta más probable. Este algoritmo está implementado en MSTParser.

El algoritmo funciona en dos etapas, que pueden observarse en la Figura 2.8, siendo la primera una etapa de dependencias sin etiquetar y la segunda de dependencias etiquetadas.

- Etapa 1: La primera etapa del sistema crea un análisis sin etiquetar y dada una frase de entrada x, usando MIRA (Crammer y Singer, 2001), un algoritmo basado en aprendizaje que es capaz de definir un rico conjunto de características sobre decisiones para el análisis.
- Etapa 2: Consiste en una etapa de etiquetado sobre la salida de la etapa 1, es decir, dado el análisis y de la frase x, se clasifica

[6]http://www.seas.upenn.edu/ strctlrn/MSTParser/MSTParser.html

cada rama *(i,j)* de y con una etiqueta particular l, que consiste en la etapa de caracter más probabilístico y que resulta una aproximación sin duda muy interesante.

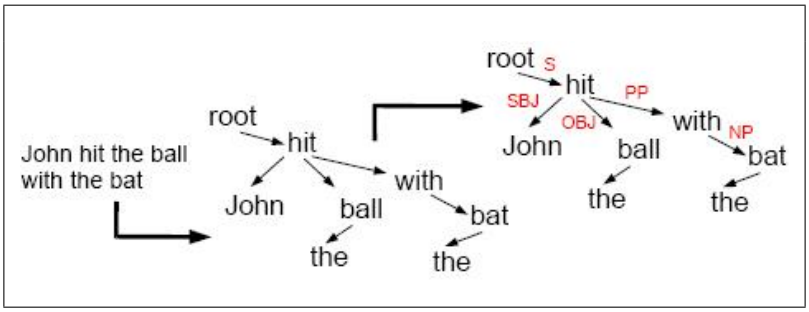

Figura 2.8: Proceso del algoritmo de McDonald et al. Las dos etapas

Esta aproximación basada en dos tipos de algoritmos y en definitiva dos etapas, obtuvo los mejores resultados. Pero dada la apertura de Maltparser para su uso y los posteriores avances que se hicieron (después de la CoNLL–X Shared Task), se extendió más el uso de Maltparser.

Formato de Archivo CoNLL (CoNLL Data Format)

El formato de archivo CoNLL Data Format surgió en la CoNLL–X Shared Task como una alternativa a otros formatos estándares como XML. Consiste en texto plano que tiene formato de matriz y en cada columna se aporta siempre el mismo tipo de información y cada fila es una palabra o token de la frase.

Como se puede observar en la Figura 2.9 y en la tabla 2.3 de la página 37 se aprecia el formato de matriz, por tanto el análisis de la frase *Inesperadamente, los límites de su vida se habían reducido a la mínima expresión* se muestra en la Figura 2.9 y la Tabla 2.3, en columna. Se observan 10 columnas cuyo objetivo se describe a continuación. Hay 2 tipos de formato CoNLL, uno que contiene 10 columnas que contienen toda la información relativa al árbol de dependencias y un formato *ciego* que sólo contiene 6 columnas y es el que se utiliza para los corpora de test, ya que no se dispone de la información relativa al árbol de dependencias.

Un archivo en formato CoNLL Data Format puede contener un número indefinido de frases separadas por una línea (fila) en blanco. Es interesante añadir que está codificado en formato UTF-8 (Unicode). Todas las palabras están etiquetadas con los siguientes 10 campos: ID, FORM, CPOSTAG, POSTAG, HEAD, DEPREL, PHEAD, PDEPREL. Si no es posible colocar ningún contenido en alguna etiqueta se usa un guión.

Este formato de archivo aporta más claridad que complejos ficheros XML

```
1       Inesperadamente Inesperadamente r       rg              _       10      CC      10      CC
2       ,       ,       F       Fc      _       1       PUNC    1       PUNC
3       los     el      d       da      gen=m|num=p     4               4
4       límites límite  n       nc      gen=m|num=p     10      SUJ     10      SUJ
5       de      de      s       sp      for=s   4               4
6       su      su      d       dp      num=s|per=3|gen=c       7               7
7       vida    vida    n       nc      num=s|gen=f     5               5
8       se      él      p       p0      per=3   10              10
9       habían  haber   v       va      num=p|per=3|mod=i|tmp=i 10              10
10      reducido        reducir v       vm      gen=m|num=s|mod=p       0       ROOT    0       ROOT
11      a       a       s       sp      for=s   10      CREG    10      CREG
12      la      el      d       da      num=s|gen=f     13              13
13      mínima  mínimo  a       aq      num=s|gen=f     11              11
14      expresión       expresión       n       nc      num=s|gen=f     13              13      _
15      .       .       F       Fp              10      PUNC    10      PUNC
```

Figura 2.9: Frase ejemplo en formato CoNLL Data Format del corpus AnCora.

Tabla 2.3: Frase ejemplo en formato CoNLL Data Format, usando el mismo ejemplo de la Figura 2.9 pero en este caso se muestra en forma de matriz, para favorecer la comprensión del lector.

ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL	PHEAD	PDEPREL
1	Inesperadamente	Inesperadamente	r	rg	–	10	CC	10	CC
2	,	,	F	Fc	–	1	PUNC	1	PUNC
3	los	el	d	da	gen=m num=p	4	–	4	–
4	límites	límite	n	nc	gen=m num=p	10	SUJ	10	SUJ
5	de	de	s	sp	for=s	4	–	4	–
6	su	su	d	dp	num=s gen=f	5	–	5	–
7	vida	vida	n	nc	num=s gen=f	5	–	5	–
8	se	el	p	p0	per=3	10	–	10	–
9	habían	haber	v	va	num=p per=3 mod=i tmp=i	10	–	10	–
10	reducido	reducir	v	vm	gen=m num=s mod=p	0	ROOT	0	ROOT
11	a	la	s	sp	for=s	10	CREG	10	CREG
12	la	el	d	da	num=s gen=f	13	–	13	–
13	mínima	mínimo	a	aq	num=s gen=f	11	–	11	–
14	expresión	expresión	n	nc	num=s gen=f	13	–	13	–
15	.	.	F	Fp	–	10	PUNC	10	PUNC

ya que las etiquetas son siempre las mismas y para el procesado mediante scripts es mucho más sencillo el tratamiento. A continuación se explica el contenido de cada campo.

- ID: Identificador de elemento, empieza en 1, para cada frase.
- FORM: Palabra o signo de puntuación.
- LEMMA: Lema de la palabra. Si no hay se usa un guión.
- CPOSTAG: Tipo de palabra general (nombre, verbo, etc).

- POSTAG: Tipo de palabra con más información (nombre común, nombre propio, etc).
- FEATS: Conjunto de características sintácas de la frase separadas por barras verticales (genero, número, etc).
- HEAD: Cabecera de la palabra, es decir, de qué palabra depende.
- DEPREL: Relación de dependencia con la cabecera.
- PHEAD: Cabecera proyectiva. En el caso de AnCora es irrelevante porque contiene la misma información que la etiqueta HEAD.
- PDEPREL: Relación de dependencia *proyectiva*. En el caso de AnCora es irrelevante porque contiene la misma información que la etiqueta DEPREL.

2.4.2. CoNLL XI Shared Task (2007)

Como en 2006, la CoNLL Shared Task, esta vez en su undécima versión (Nivre et al., 2007) tuvo como tema central el análisis de dependencias automático, pero añadiendo el problema de la adaptación al dominio.

Se hizo de la siguiente manera: por un lado se mantuvo el mismo esquema que en la anterior Shared Task añadiendo algunas lenguas nuevas. Y por el otro lado, una tarea de adaptación al dominio, donde la tarea fue usar una máquina de aprendizaje para adaptar un analizador desde un lenguaje simple, a un nuevo dominio. Hubo veintitrés sistemas para la primera tarea y diez para la segunda. Hubo un total de veintiún artículos admitidos, algunos de los cuales describia más de un sistema o solamente uno.

Para el proyecto que nos ocupa nos interesa la primera tarea de investigación, que fue la misma que se propuso en la CoNLL X Shared Task de 2006. Al igual que en 2006, el objetivo era medir la calidad en el análisis. Una de las ventajas del análisis de dependencias en contraposición al análisis de constituyentes es que se adapta automáticamente a las lenguas con un orden libre de las palabras (free-word order languages). Una de las conclusiones más importantes obtenidas en la ConLL X Shared Task de 2006 fue que la calidad en el análisis difería profundamente dependiendo de que idioma se estuviese analizando y se marcó como algo importante para la investigación futura; por tanto en esta nueva reunión se decidió añadir nuevas lenguas y mantener también las anteriores para observar si se siguió la tendencia de 2006. Los corpus de entrenamiento ofrecidos tenían, en la medida de lo posible, el mismo formato que los originales. Se estableció un tamaño máximo para los corpora para que no faltase tiempo para el entrenamiento.

Las lenguas que formaron parte finalmente fueron:árabe, catalán, chino, checo, inglés, griego, húngaro, italiano, turco y vasco. Como puede verse en esta convocatoria no se tuvo en cuenta el castellano, por lo que el estudio de este tipo de sistemas se centra en la convocatoria de 2006, la CoNLL–X Shared Task.

2.5. Recapitulación

En este Capítulo se ha presentado el Análisis Sintáctico de Dependencias, las diferentes aproximaciones existentes actualmente para el análisis y generación de analizadores, así como corpora de entrenamiento y prueba. Se justifica la elección del generador de analizadores de dependencias (Maltparser) y el corpus utilizado en este trabajo (AnCora). Además se presentan las CoNLL Shared Tasks de 2006 y 2007, que fijaron y centraron la investigación del momento en el análisis sintáctico automático y multilingüe de dependencias.

Capítulo 3

Hacia una Mejor Precisión en el Análisis Sintáctico de Dependencias para el Castellano

En este apartado de la memoria, se presentan los distintos experimentos de investigación realizados durante el desarrollo del trabajo. En primer lugar se exponen una serie de técnicas de bajo nivel para evaluar la variación de la calidad en el análisis, variando el tamaño de los corpora de entrenamiento. Posteriormente, se presentan técnicas de más bajo nivel donde se expone en primer lugar una breve aproximación hacia un analizador de n-versiones, centrándose en el análisis de la conjunción, y posteriormente para el resto de palabras que peor resultado ofrecen tras el análisis y que por tanto son significativas para obtener un mejor análisis.
En este trabajo, el primer paso fue replicar el experimento realizado por Nivre en la CoNLL–X Shared Task. Entrenamos Maltparser 0.4 con la sección de AnCora dada como corpus de entrenamiento en la ConLL–X Shared Task(89.334 palabras) y las opciones de análisis y entrenamientos las propuestas por el grupo de Nivre (Nivre et al., 2006). Una vez que tuvimos un modelo entrenado, lo usamos para analizar el corpus de test (5.694 palabras), obteniendo los siguientes resultados LAS = 81,30 %, UAS = 84,67 % y LA = 90,06 %. Estos resultados nos han servido como punto de partida para los experimentos realizados en este trabajo y que se exponen a continuación.

3.1. Técnicas de Alto Nivel hacia la Mejora del Análisis para el Castellano.

En esta sección se exponen una serie de técnicas de alto nivel para el estudio de la homogeneidad en el análisis, buscando la mejora del mismo, y centrando los esfuerzos en el tamaño del corpus de entrenamiento y la longitud de las frases contenidas en los corpora de análisis y entrenamiento.

3.1.1. Hipótesis

Partimos de los resultados obtenidos en la CoNLL X Shared Task (Buchholz y Marsi, 2006), por los distintos grupos para el castellano. El objetivo del experimento que se propone es encontrar estrategias para obtener un mejor análisis cuando el analizador (basado en aprendizaje) no puede ser modificado. Se trató de enfocar el trabajo sobre como obtener un mejor análisis basándose en las ideas de (Nivre et al., 2006) y (Herrera y Gervás, 2008). Se hicieron algunos experimentos donde se estudió el efecto del tamaño del corpus de entrenamiento y la longitud de las frases en la precisión de análisis.

Se plantean las siguientes hipótesis:

- Un analizador produce una precisión de análisis homogénea.
- No es tan importante el tamaño del corpus de entrenamiento como la adecuación de los ejemplos que lo componen. Aún así el tamaño del corpus es relevante.
- La longitud de las frases del corpus de entrenamiento es relevantes a la hora de obtener un mejor análisis.

En las siguientes subsecciones se comprueban o refutan las anteriores hipótesis.

3.1.2. Homogeneidad de la Precisión de un Analizador.

Después de repetir los buenos resultados obtenidos por el grupo de Nivre en la CoNLL-X Shared Task, repitiendo la Tarea allí propuesta con la misma configuración de Maltparser, nos hicimos la siguiente pregunta: ¿Podemos esperar los mismos resultados para cualquier texto analizado con un modelo entrenado con Maltparser 0.4? Para encontrar una respuesta a esta pregunta realizamos el siguiente experimento que se muestra a continuación.

Configuración del experimento.

En primer lugar, dividimos el corpus AnCora en 21 subcorpora de aproximadamente 4.500 palabras cada uno. Distribuimos las frases de forma homogénea en los 21 subcorpora de acuerdo con la longitud de las mismas. Por

tanto, cada subcorpus contenía un número similar de frases para cada longitud. Entonces, entrenamos un modelo con Maltparser 0.4 (con la misma parametrización que la propuesta por el grupo de Nivre para el castellano en la CoNLL-X Shared Task) con cada subcorpus. De modo que obtuvimos 21 modelos diferentes preparados para el análisis. Con cada modelo, analizamos los 20 subcorpora restantes que no sirvieron para entrenar ese modelo. De esta forma obtuvimos $21 \times 20 = 420$ corpora que evaluamos. Como medidas de evaluación no sólo computamos LAS, UAS, y LA, sino también las siguientes que consideramos que serían útiles para encontrar respuestas.

- $r_{LAS,UAS}$: El coeficiente de correlación entre los obtenidos por el modelo en las 20 evaluaciones. Indica si existe una correlación entre LAS y UAS independientemente del texto analizado.
- $r_{LAS,LA}$: El coeficiente de correlación entre los obtenido por el modelo en las 20 evaluaciones. Indica si existe una correlación entre LAS y LA independientemente del texto analizado.
- $r_{UAS,LA}$: El coeficiente de correlación entre los obtenidos por el modelo en las 20 evaluaciones. Indica si existe una correlación entre LAS y LA independientemente del texto analizado.
- El máximo LAS (max_{LAS}) y el mínimo LAS(min_{LAS}) obtenidos por el modelo en las 20 evaluaciones. Estos valores dan una idea de la variación de LAS cuando se analizan diferentes textos.
- El máximo UAS (max_{UAS}) y el mínimo UAS (min_{UAS}) obtenidos por el modelo en las 20 evaluaciones. Estos valores dan una idea de la variación de UAS cuando se analizan diferentes textos.
- El máximo LA (max_{LA}) y el mínimo LA (min_{LA}) obtenidos por el modelo en las 20 evaluaciones. Estos valores dan una idea de la variación de LA cuando se analizan diferentes textos.

Los resultados pueden observarse de manera general en las tablas 3.1 y 3.2.

Resultados del experimento

Las tablas 3.1 y 3.2 muestran los resultados de la evaluación de los 20 corpora analizados considerando coeficientes de correlación en la primera tabla y valores mínimos y máximos en la segunda tabla. Las figuras 3.1, 3.2 y 3.3 muestran en forma de nube de puntos la correlación entre las distintas medidas de evaluación.

Si analizamos la segunda columna de la tabla ($r_{LAS,UAS}$), podemos concluir que la correlación entre LAS y UAS es elevada. Podemos tomar la

Tabla 3.1: Resultados obtenidos pos los modelos entrenados con los 21 subcorpora en que dividimos AnCora. Considerando los coeficientes de correlación entre las medidas LAS, UAS y LA tomadas por pares.

Subcorpus	$r_{LAS,UAS}$	$r_{LAS,LA}$	$r_{UAS,LA}$
A_0	0,84	0,78	0,44
A_1	0,87	0,85	0,65
A_2	0,92	0,87	0,91
A_3	0,92	0,80	0,54
A_4	0,88	0,87	0,71
A_5	0,82	0,89	0,61
A_6	0,90	0,88	0,69
A_7	0,86	0,88	0,63
A_8	0,94	0,87	0,71
A_9	0,91	0,76	0,56
A_{10}	0,87	0,89	0,69
A_{11}	0,91	0,85	0,69
A_{12}	0,92	0,78	0,60
A_{13}	0,89	0,85	0,64
A_{14}	0,92	0,81	0,62
A_{15}	0,94	0,85	0,76
A_{16}	0,87	0,76	0,44
A_{17}	0,92	0,79	0,58
A_{18}	0,95	0,78	0,63
A_{19}	0,82	0,86	0,54
A_{20}	0,95	0,84	0,73
max	0,95	0,89	0,91
avg	0,90	0,83	0,64
min	0,82	0,76	0,44

misma conclusión si observamos la columna $r_{LAS,LA}$. Estos valores son lógicos considerando la definición de LAS, LA y UAS. LAS depende de un buen etiquetado de la relación de dependencia y también de la decisión de que palabra depende. La decisión de que de que palabra depende afecta directamente a UAS, y la relación de dependencia afecta a LA. Pero cuando analizamos las cuarta columna ($r_{UAS,LA}$), observamos muy distintos valores para $r_{UAS,LA}$, desde 0.44 a 0.91, lo que puede también explicarse teniendo en cuenta las definiciones de UAS y LA, ya que son medidas que tienen relación pero no directa.

Si consideramos los 21 modelos, el que da una máxima variación en sus series de LAS es A_8 con una diferencia de 5,21 puntos. El que muestra una mínima variación entre sus series de LAS es A_4 con una diferencia de 3,06 puntos. Para UAS, la variación máxima está dada por A_2 con una diferencia de 9,43 puntos, y A_1 que muestra la mínima variación con una diferencia de 2,41 puntos. Finalmente, A_2 alcanza la máxima variación no sólo por UAS pero también para LA (10,61 puntos), mientras la mínima variación ocurre otra vez con el modelo A_4 con 2,26 puntos que muestra la mínima variación para LAS. Partiendo de estos valores podemos concluir que cada modelo se

Tabla 3.2: Resultados obtenidos pos los modelos entrenados con los 21 subcorpora en que dividimos AnCora considerando valores máximos y mínimos de las medidas LAS, UAS y LA.

Subcorpus	max_{LAS}	min_{LAS}	max_{UAS}	min_{UAS}	max_{LA}	min_{LA}
A_0	72,78 %	69,06 %	78,22 %	74,20 %	85,03 %	82,45 %
A_1	72,86 %	68,81 %	**77,29 %**	**74,88 %**	84,85 %	82,36 %
A_2	73,55 %	68,40 %	**78,47 %**	**69,04 %**	**85,85 %**	**75,24 %**
A_3	72,68 %	69,01 %	77,58 %	74,42 %	85,39 %	82,35 %
A_4	**72,02 %**	**68,96 %**	77,09 %	74,31 %	**84,71 %**	**82,45 %**
A_5	72,74 %	69,32 %	77,32 %	74,90 %	84,99 %	82,40 %
A_6	71,90 %	68,42 %	76,88 %	74,32 %	84,78 %	82,20 %
A_7	72,55 %	68,16 %	77,27 %	73,61 %	85,24 %	81,95 %
A_8	**72,92 %**	**67,71 %**	77,55 %	73,65 %	85,52 %	82,03 %
A_9	72,27 %	68,23 %	77,47 %	73,99 %	84,85 %	82,35 %
A_{10}	71,76 %	68,19 %	77,11 %	73,00 %	84,62 %	81,74 %
A_{11}	73,30 %	68,37 %	78,27 %	73,68 %	85,73 %	82,52 %
A_{12}	73,01 %	69,27 %	78,32 %	74,62 %	85,39 %	82,43 %
A_{13}	72,96 %	69,61 %	78,22 %	74,46 %	85,60 %	82,19 %
A_{14}	73,04 %	68,29 %	77,93 %	74,07 %	85,04 %	82,27 %
A_{15}	71,37 %	67,81 %	76,21 %	72,60 %	85,01 %	82,42 %
A_{16}	72,51 %	68,83 %	76,83 %	73,89 %	85,50 %	82,17 %
A_{17}	73,23 %	68,82 %	77,58 %	74,17 %	85,78 %	82,77 %
A_{18}	72,50 %	67,40 %	77,63 %	73,18 %	84,70 %	81,82 %
A_{19}	72,25 %	68,99 %	77,65 %	74,19 %	85,39 %	82,99 %
A_{20}	72,73 %	68,28 %	77,55 %	73,68 %	85,19 %	82,07 %
max	**73,55 %**	**69,61 %**	**78,47 %**	**74,90 %**	**85,85 %**	**82,99 %**
avg	**72,62 %**	**68,57 %**	**77,54 %**	**73,76 %**	**85,20 %**	**81,96 %**
min	**71,37 %**	**67,40 %**	**76,21 %**	**69,04 %**	**84,62 %**	**75,24 %**

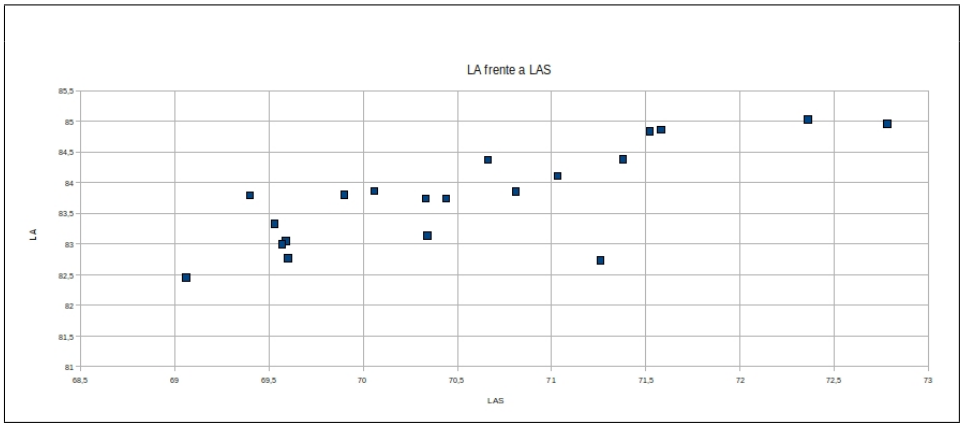

Figura 3.1: Correlación en forma de nube de puntos entre LA y LAS.

puede mover en un rango de precisión dependiendo de los textos que son usados como entrada. Pero el resultado global, considerando los 21 modelos, es más homogeneo, como puede verse en las tres últimas filas de la Tabla 3.1, desde la quinta columna, hasta el final.

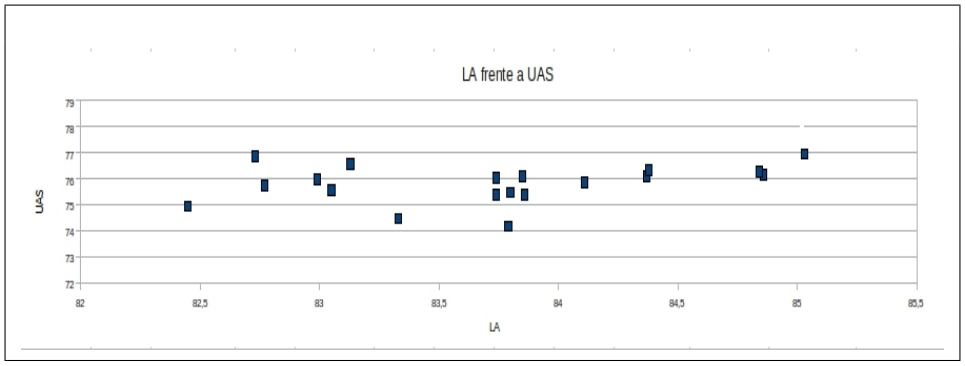

Figura 3.2: Correlación en forma de nube de puntos entre LA y UAS.

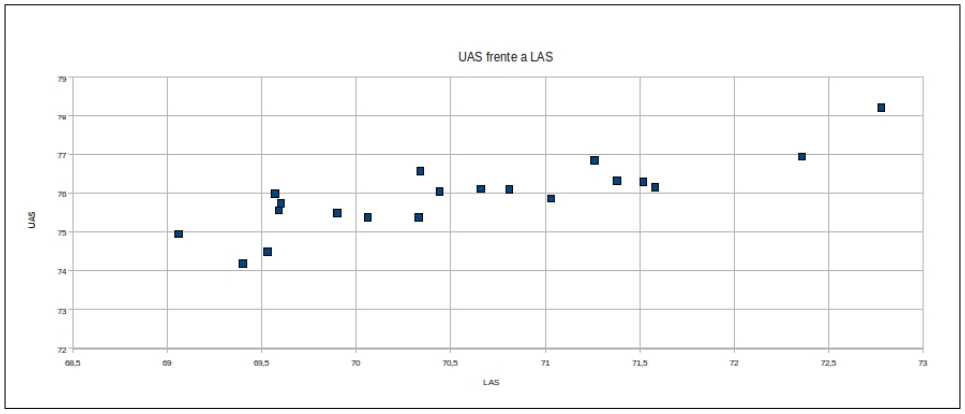

Figura 3.3: Correlación en forma de nube de puntos entre UAS y LAS.

Conclusiones del experimento

Como conclusión podemos decir que entrenando dos modelos con corpora diferentes (con tamaño similar y con un número similar de frases) deberían dar un resultado global similar. Sin embargo, cada modelo puede mostrar probablemente diferentes valores de precisión dependiendo del texto específico usado como entrada. Esto puede explicarse porque el tamaño del copus y la longitud de sus frases puede influir a la calidad del análisis. Tomando estos resultados como base, se realizaron los experimentos que se muestran en las siguientes subsecciones. Los resultados obtenidos aquí dan paso a una evaluación más compleja de los resultados, teniendo en cuenta también la establidad de la precisión y no solo lo bien que se ha analizado en términos globales.

Basándose en las distintos análisis de un mismo subcorpus con diferentes modelos, realizamos un último estudio usando los datos obtenidos por los 420 entrenamientos. Si cada A_i es analizado con un modelo entrenado por $A_j, j \neq i$ nos permitirá obtener el mejor valor posible de LAS de A_i, y el

mejor resultado global de LAS combinando la acción de todos los modelos. Es decir, se pretende descubrir qué modelo entrenado por un corpus específico es mejor para analizar otro corpus específico. La tabla 3.3 muestra cómo cada modelo puede usarse para analizar cada A_i.

Tabla 3.3: Modelos entrenados con $A_j, j \neq i$ que deben ser usados para analizar A_i para obtener el mejor valor de LAS posible.

Entrada	**Analizado por el modelo**	**LAS**
A_0	A_{14}	72,58 %
A_1	A_{18}	72,20 %
A_2	A_{17}	70,93 %
A_3	A_1	70,99 %
A_4	A_{12}	72,47 %
A_5	A_{13}	70,17 %
A_6	A_2	73,55 %
A_7	A_0	71,58 %
A_8	A_4	71,57 %
A_9	A_{17}	73,23 %
A_{10}	A_3	71,91 %
A_{11}	A_3	71,78 %
A_{12}	A_3	71,27 %
A_{13}	A_7	69,83 %
A_{14}	A_6	71,90 %
A_{15}	A_5	70,24 %
A_{16}	A_5	70,07 %
A_{17}	A_1	70,77 %
A_{18}	A_{12}	71,94 %
A_{19}	A_{17}	71,40 %
A_{20}	A_{17}	70,79 %
LAS medio		**71,48 %**

La obtención de estos resultados se extrajo del experimento 3.1.2, buscando aquellos analizadores que mejor se habían comportado para cada subcorpus. De esta manera se puede observar que hay ciertos modelos que son capaces de analizar mejor una cantidad mayor de corpora, mientras que hay modelos que no son capaces de analizar mejor ningún corpus.

Estas ideas invitan a la generación de corpora menos ambiguo en algunos casos y a la eliminación de ciertas apariciones de frases presentes en los corpora utilizados para entrenar los modelos menos competitivos. De esta manera, y tras un minucioso proceso manual de verificación se puede obtener un mejor análisis. Por otro lado, la idea de generar un algoritmo capaz de utilizar cada modelo según sea el caso, o la posible selección posterior del

mejor analizador para cada frase o corpus. Ideas similares se desarrollan en la Sección 3.2, en (Ballesteros et al., 2010a), (Ballesteros et al., 2010c) y (McDonald y Nivre, 2007).

3.1.3. Relación entre el Tamaño del Corpus y la Precisión en el Análisis.

Este nuevo experimento se centra en el efecto del tamaño del corpus sobre la precisión en el análisis. Para analizar este efecto se fue construyendo un corpus de entrenamiento incremental, se entrenó un analizador concreto para cada incremento y se evaluó la calidad del análisis en cada iteración.

Configuración del experimento

El metodo que se siguió para realizar este experimento fue el siguiente:

- En primer lugar seleccionamos los $A_i, 1 \leq i \leq 20$, para los cuales se obtuvo mejor LAS cuando se analizaron con el modelo entrenado por A_0 en el experimento anterior. Este subcorpus fue A_6 y este fue el primer subcorpus añadido al corpus incremental.

- En cada iteración entrenamos Maltparser 0.4 con el corpus incremental.

- En cada iteración comprobamos la precisión del modelo entrenado analizando con el A_0.

- En cada iteración se añade al corpus incremental el A_i no usado para el cual se ha obtenido mejor LAS de los restantes cuando se analiza por el modelo entrenado con A_0.

- Iteramos 20 veces hasta que todos los A_i hayan sido añadidos al corpus incremental.

En cada iteración el corpus mantiene la proporción de frases teniendo en cuenta sus longitudes.

Resultados del experimento

Los resultados de este experimento pueden verse en la Figura 3.4, donde se pueden observar los valores para LAS, UAS, y LA para cada iteración. En el eje x se representa el número de palabras contenido en el corpus incremental.

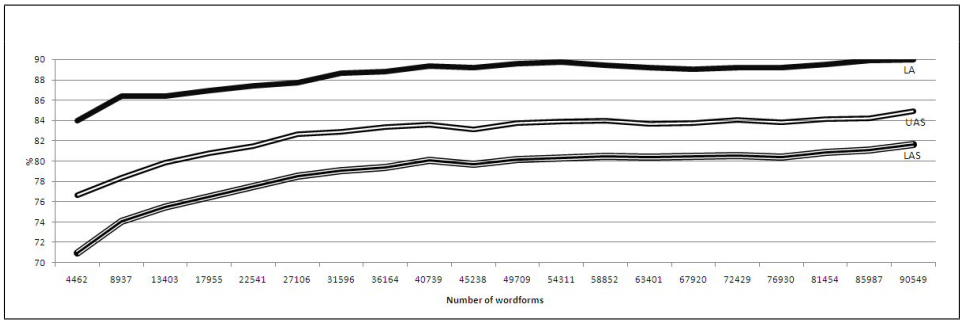

Figura 3.4: LAS, UAS y LA dependiendo del número de palabras contenidas en el corpus de entrenamiento.

Conclusiones del experimento

Considerando LAS, desde la primera a la segunda iteración es casi 3 puntos mayor. Desde la segunda a la tercera iteración LAS se incrementa 1,38 puntos. En la cuarta iteración LAS es 1,2 puntos mayor. Y crece casi 1 punto en cada una de la quinta y sexta iteración. Añadiendo unas 22.600 palabras al corpus de entrenamiento se obtiene un incremento de LAS de 7,56 puntos. Pero añadiendo otras 22.600 palabras se obtiene un incremento de sólo 1,63 puntos. UAS y LA muestran un comportamiento similar a LAS.

3.1.4. Relación entre la Longitud de la Frase y la Precisión del Analizador.

Fijándose en (McDonald y Nivre, 2007) –donde los errores en el análisis están relacionados con la longitud de las frases– y (Herrera y Gervás, 2008) –donde el efecto de las longitud de las frases para la precisión en el análisis es estudiado– el otro parámetro donde nos fijamos para el estudio fue la longitud de las frases. Se realizaron 2 experimentos para determinar si la longitud de las frases podrían afectar a la precisión en el análisis.

Experimento I

El objetivo de este primer experimento consistió en comprobar si la precisión en el análisis variaba según la longitud de las frases contenidas en el corpus de entrenamiento. Para ello, se generó una serie de corpora partiendo del corpus de entrenamiento original y se procedió como se muestra a continuación.

Configuración del experimento I

Como primera prueba, dividimos la sección de AnCora que fue usada como corpus de entrenamiento en la CoNLL–X Shared Task en 102 subcor-

pora, cada uno conteniendo frases de longitud única. Es decir un corpus de 1 frase de 143 palabras, otro subcorpus con 1 frase de 130 palabras, otro subcorpus de 2 frases de 128 palabras cada una y así sucesivamente.

Resultados del Experimento I

Entrenando Maltparser 0.4 con cada uno de esos subcorpora, obtuvimos los resultados que se muestran en la Figura 3.5, en la página 51. En el eje x se representa la longitud de las frases contenidas en el subcorpus usado como corpus de entrenamiento, y en el eje y la precisión obtenida, utilizando las 3 medidas de evaluación, en cada iteración. La gráfica interna de la figura 3.5 muestra el número de frases de cada longitud presentes en el corpus AnCora, que se muestra ampliado en la Figura 3.6. Se incluye, para observar como varían los resultados en relación a la cantidad de frases presentes de cada longitud.

Conclusiones del Experimento I

Como puede observarse, los corpora de entrenamiento que contienen frases de gran longitud producen alta precisión a pesar del pequeño tamaño de los corpora en número de palabras. Por ejemplo, un corpus de 143 palabras, con una 'unica frase produce: 45,88 % LAS, 51,16 % UAS y 66,15 % LA. AnCora tiene 35 frases, con 80 o más de 80 palabras. La figura 3.5 muestra que las frases largas son una pequeña parte del corpus. ¿Podemos conseguir buenos resultados, considerando sólo frases largas?.

Experimento II

En este experimento se intenta comparar la precisión en el análisis de modelos entrenados con corpora que contiene tan sólo frases largas con modelos entrenados con corpora que sólo tiene frases cortas. Buscando que el tamaño de los corpora de entrenamiento resultantes sean del mismo tamaño.

Configuración del Experimento II

Para verificar y contestar a la pregunta formulada en el anterior experimento, se añadieron sistemáticamente frases creando nuevos subcorpora de entrenamiento con frases largas y entrenamos un modelo con cada subcorpus. Empezando con frases de 120 palabras o más, depués repetimos con frases de 110 palabras o más, seguidamente con frases de 100 palabras o más, después con frases de 90 palabras o más y finalmente con frases de 80 palabras o más.

Para comparar estos resultados se repitió el experimento, pero sólo con frases de poca longitud: 10 palabras o menos. Comenzando en primer lugar

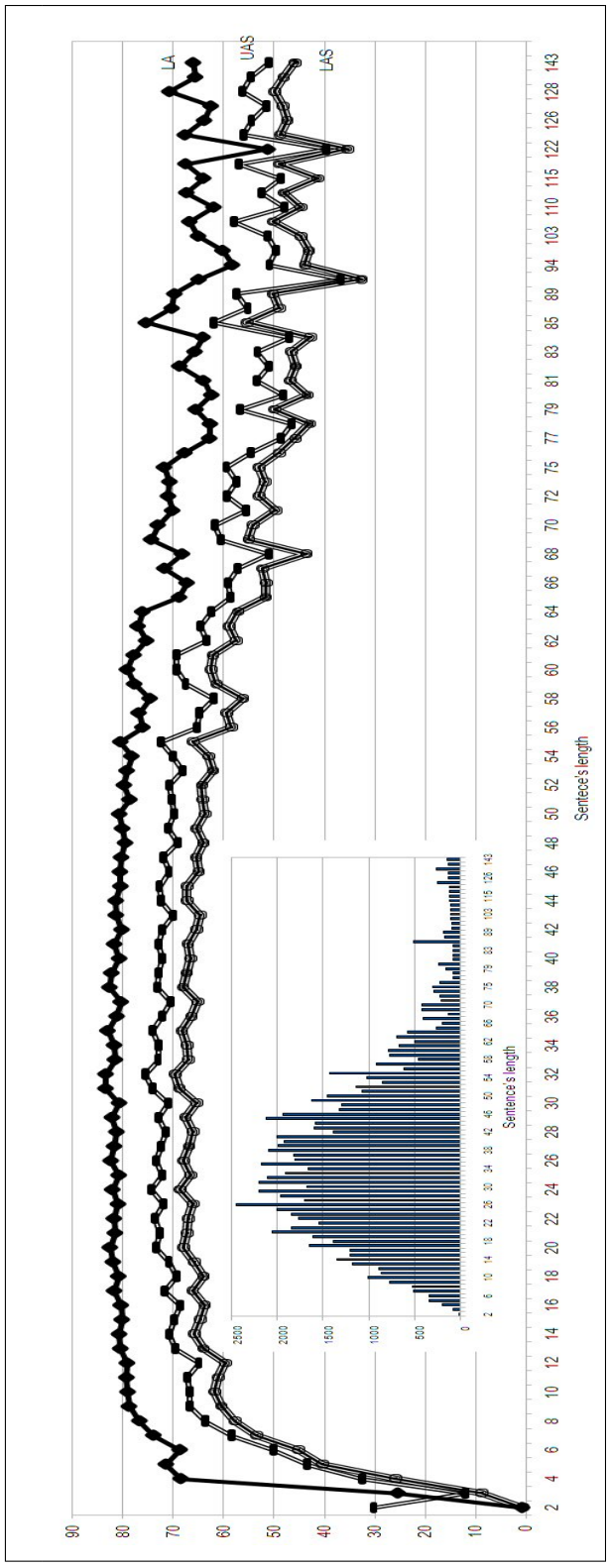

Figura 3.5: LAS, UAS y LA cuando se entrena con corpora de frases de igual tamaño. En la gráfica interna se observa el número de frases de cada longitud, y en la gráfica externa se observa la calidad obtenida por cada modelo entrenado con cada subcorpus.

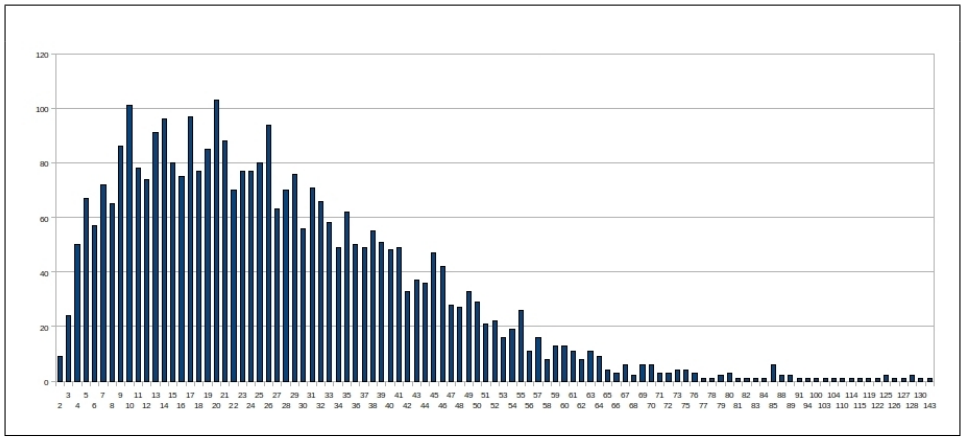

Figura 3.6: Número de frases de cada longitud presentes en el corpus AnCora. La frase que contiene más palabras contiene 143 y la que menos tan sólo 2.

con frases de 7 palabras o menos, posteriormente con frases de 8 palabras o menos y así sucesivamente.

Resultados del Experimento II

La tabla 3.4 muestra LAS, UAS y LA para esos subcorpora, formados por frases de mucha longitud. Los resultados son notablemente altos, comparando con corpora de más de 4500 palabras y que contienen frases de todos los tamaños. Comparativamente tenemos los resultados de la tabla 3.5 donde se observa que con corpora de mayor tamaño se obtienen resultados de peor calidad que en el experimento de las frases largas.

Tabla 3.4: Resultados obtenidos por los modelos entrenados con los subcorpora, sólo frases largas.

Subcorpus	Tamaño(palabras)	LAS	UAS	LA
120..143	1154	61,07 %	67,60 %	77,18 %
110..143	1612	64,64 %	70,23 %	80,24 %
100..143	1919	66,68 %	71,99 %	81,41 %
90..143	2104	66,86 %	72,17 %	81,90 %
80..143	3538	68,60 %	74,11 %	82,72 %

Conclusiones del Experimento II

Partiendo de estos resultados, se puede concluir que los corpora de frases largas contribuyen a un mejor resultado global. Además se observa que

Tabla 3.5: Resultados obtenidos por los modelos entrenados con los subcorpora, sólo frases cortas.

Subcorpus	Size(wordforms)	LAS	UAS	LA
2..7	1471	57,60 %	62,19 %	78,10 %
2..8	1991	62,16 %	66,86 %	79,75 %
2..9	2765	64,51 %	68,90 %	81,52 %
2..10	3775	66,84 %	72,30 %	82,79 %

un corpus de entrenamiento de frases largas necesita menos palabras para obtener resultados similares que un corpus que contiene sólo frases cortas.

Experimento III

Este tercer experimento consistió en descubrir hasta que punto era necesario incluir las frases de los corpora cuyos modelos entrenados tuvieron peores resultados en el Experimento I, para ello se fuá añadiendo incrementalmente las frases a un único corpus de entrenamiento, de esa manera se consiguió generar modelos capaces de analizar las frases del corpus de test.

Configuración del Experimento III

Después realizamos otro experimento centrándonos sólo en la longitud de las frases. Para ello construimos incrementalmente un corpus de entrenamiento y evaluamos el resultado para cada modelo entrenado. El metodo fue el siguiente:

- En primer lugar seleccionamos el corpus para el cual se obtuvo mejor LAS cuando se analizaron en el experimento de la Sección 3.1.3.
- En cada iteración entrenamos Maltparser 0.4 con el corpus incremental.
- En cada iteración comprobamos la precisión analizando el corpus de test de la CoNLL–X Shared Task.
- En cada iteración se añade al corpus incremental el subcorpus no usado para el cual se ha obtenido mejor LAS de los restantes cuando se analizó en el experimento previo
- Iteramos hasta que se consume todo el corpus de entrenamiento de la CoNLL–X Shared Task.

Resultados del Experimento III

La figura 3.7 muestra LAS, UAS y LA para cada análisis. En el eje x, representamos el número de palabras contenidas en el corpus incremental, y en el eje y la precisión obtenida en cada iteración.

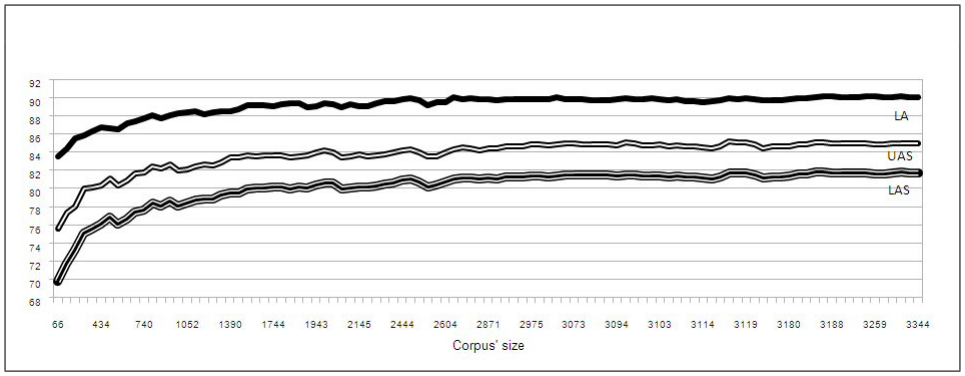

Figura 3.7: LAS, UAS y LA cuando se considera la longitud de las frases para construir el corpus de entrenamiento.

Conclusiones del Experimento III

Partiendo de los resultados de la Figura 3.7, podemos concluir, que la selección del corpus de entrenamiento basándose en la longitud de las frases permite obtener mejores resultados globales, por lo tanto es interesante seleccionar la mayor parte de las veces frases de elevada longitud y de esta manera se obtendrá un análisis mejor. Es necesaria la existencia de frases cortas, pero no en la misma proporción que las frases largas.

3.2. Técnicas de Bajo Nivel para la Mejora del Análisis para el castellano: Hacia un Analizador de N versiones.

Partiendo de los resultados obtenidos en el experimento anterior, tomamos como linea base los resultados de la CoNLL–X Shared Task (Buchholz y Marsi, 2006) de los distintos grupos para el castellano. El objetivo del experimento que se expone es demostrar que distintos analizadores específicos combinados con el uso de un analizador general pueden mejorar la precisión local y global ya que hay un conjunto de palabras que provocan el mayor número de fallos y generando analizadores específicos capaces de etiquetar mejor esas palabras.

3.2.1. Hipótesis

Dado que existe un conjunto de palabras que provocan un mayor número de errores en el análisis, ¿podemos reducir la cantidad de errores provocados por las mismas entrenando analizadores específicos y analizando las palabras problemáticas con ellas.

Considerando el experimento base y los resultados obtenidos en la CoNLL-X Shared Task, donde los resultados con carácter global son muy buenos alcanzando valores cercanos al 90 % de precisión, se observa que solo 358 palabras del corpus de test obtienen un 100 % de precisión en el análisis, lo que es un 6,3 %. Si consideramos frases, en vez de palabras, observamos que tan sólo 38 frases están perfectamente analizadas, lo que supone un 18,4 %. El usuario final espera que el análisis obtenido por el analizador de dependencias generado sea satisfactorio, pero es altamente improbable que el árbol de dependencias que obtenga esté perfectamente analizado. La mayor parte de las frases tras ser analizadas con Maltparser, muestran al menos un error tras el análisis. Como se describe en (Ballesteros et al., 2010a) y (Ballesteros et al., 2010c) cuando se observan los resultados después del análisis, se encuentra que hay un pequeño grupo de palabras que muestran la mayor parte de los errores, tanto en el etiquetado, como en la conexión con las palabras de las que dependen en el árbol de dependencias. Esas palabras son las preposiciones 'a', 'de', 'en', 'con', 'por', la conjunción 'y' o 'e' y el nexo 'que'. Como se intuye estas palabras provocan una alta ambigüedad a la hora de decidir como deben ser analizadas, ya que existen en el corpus de entrenamiento varios patrones de etiquetado para las mismas. Estas palabras causan, habitualmente, error en la relación de dependencia, en la decisión de como debe conectarse con otras palabras en el árbol o en ambas. Hay 20 frases (340 palabras) en el corpus de test con un solo error, lo que supone un 9,7 % del total de frases que contiene el corpus o un 5,98 % si consideramos las palabras. En 10 de esas 20 frases el error es provocado por alguna de las palabras mencionadas anteriormente.

Como se deduce de los datos que se exponen, un usuario final que desee un análisis perfecto no puede quedar satisfecho con esos resultados. Para algunos propositos, no es relevante que el resultado global considerando el corpus completo sea muy bueno, sin embargo, la precisión local, si consideramos la calidad en el análisis frase a frase, si lo es. El 81,6 % de las frases que se analizan individualmente contienen al menos un error. consideramos que se debe obtener un mayor número de frases que obtengan 100 % LAS después del análisis, por tanto, creemos que los esfuerzos deben centrarse en la búsqueda de técnicas que así lo consigan. Por otro lado, si mejoramos la precisión local, acabaremos obteniendo una mejor precisión global y llegar al objetivo final que es obtener un análisis de dependencias perfecto.

Nuestra hipótesis es que mejorando la precisión local, además de mejorar

la precisión global, el usuario final del analizador de dependencias quedará más satisfecho, ya que será menos probable que las frases contengan errores, o bien, que los errores contenidos tras el análisis sean menos importantes. Se llevaron a cabo una serie de experimentos para confirmar o refutar la hipótesis que se presentan en las siguientes subsecciones.

3.2.2. El Desarrollo de N Analizadores Específicos

La idea básica consiste en desarrollar un estudio sobre que mejora se puede obtener, analizando las palabras que más errores producen. El estudio consiste en encontrar el conjunto de diferentes formas de etiquetado que tienen esas palabras específicas y entrenar un analizador de dependencias específico para cada caso. Se empezó en el estudio con la conjunción y la preposición 'a', para demostrar la viabilidad de un analizador N-versiones. Se encontraron cuatro formas significativas y diferentes en la manera que la conjunción es etiquetada en los árboles de dependencia resultantes y seis casos diferentes para la preposición 'a'. De manera que para demostrar la viabilidad del estudio, se entrenaron 10 analizadores de dependencias específicos para cubrir todos los casos posibles teniendo en cuenta esas 2 palabras. Después de ello se combinó el resultado obtenido por el analizador general con los resultados obtenidos por los analizadores específicos, intercambiando los nodos concretos para los que el analizador específico se comportaba correctamente. De esta manera, cuando se analiza, por ejemplo, una conjunción, la salida del analizador general se ignora y se intercambia por la salida del analizador específico. Los resultados obtenidos en las pruebas preliminares, fueron muy buenos, lo que fue un estímulo para seguir experiementando.

Como se puede ver en las Figuras 3.8, 3.9 y 3.10, el analizador n-versiones funcionaría de la siguiente manera: el analizador general analizaría la frase completa, pudiendo cometer errores o no. Si se detecta el patrón correspondiente a un analizador específico (como en la Figura3.9), el analizador específico realiza el análisis como si fuera un analizador general. Finalmente como se ve en la figura 3.10, se intercambiaría el nodo obtenido para la palabra concreta, en este caso la preposición 'a', se ignoraría el resto del árbol que devuelve el analizador específico y se obtendría, en términos generales y locales, una frase mejor analizada. Para el ejemplo, una frase con 100 % de precisión en el análisis.

3.2.3. La Conjunción

En esta sección se explican dos aproximaciones diferentes que se aplicaron como estudios previos para ver la viabilidad del estudio del analizador de N-versiones y se tomó como banco de pruebas la conjunción, que para el castellano tiene 2 formas diferentes: 'y' y 'e'. La primera aproximación es un estudio incremental con un ejemplo concreto y la segunda la descripción

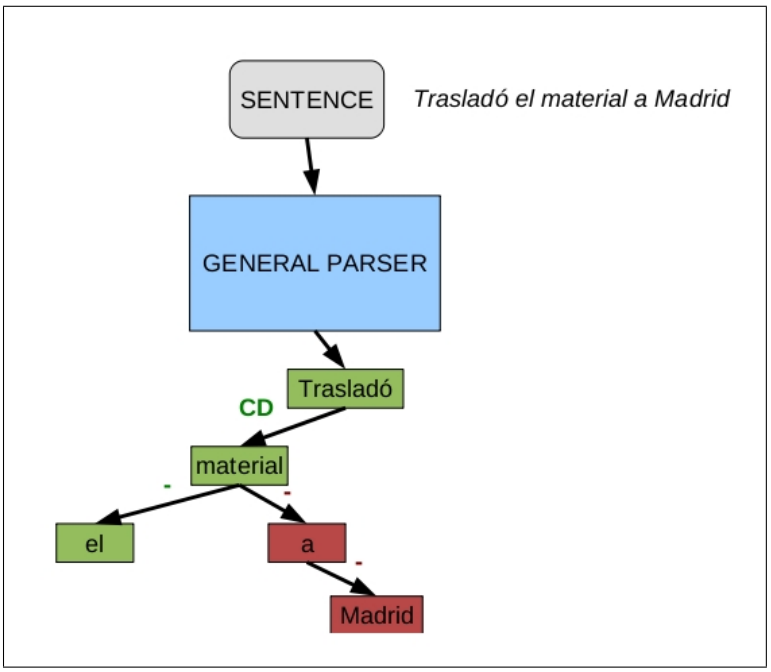

Figura 3.8: Ejemplo de la frase 'Trasladó el material a Madrid' incorrectamente analizada por el analizador general.

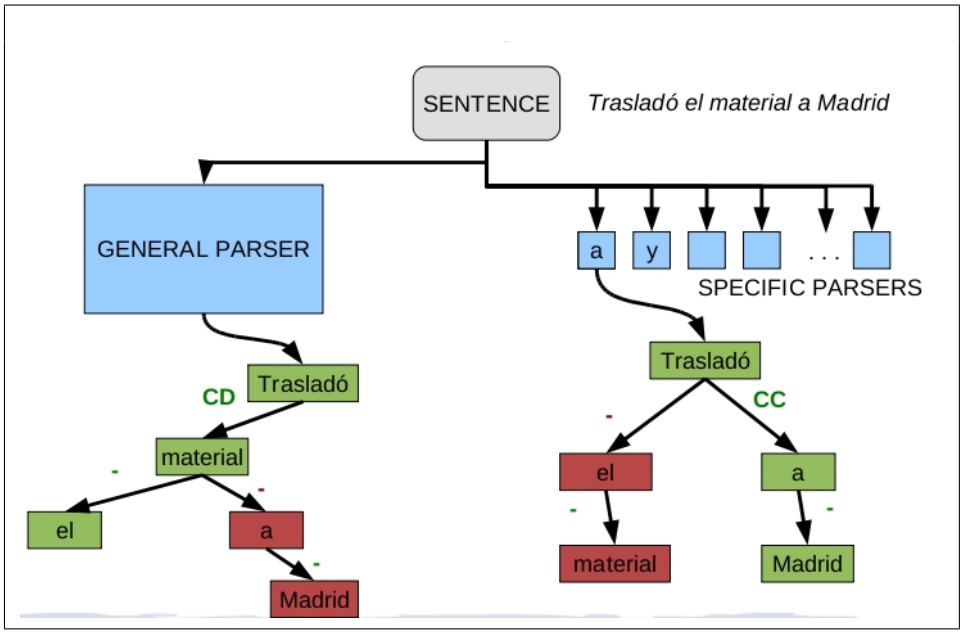

Figura 3.9: Ejemplo de la frase 'Trasladó el material a Madrid' incorrectamente analizada por el analizador general y por el analizador específico. Pero en este caso el analizador específico realiza correctamente la parte para la que ha sido entrenado.

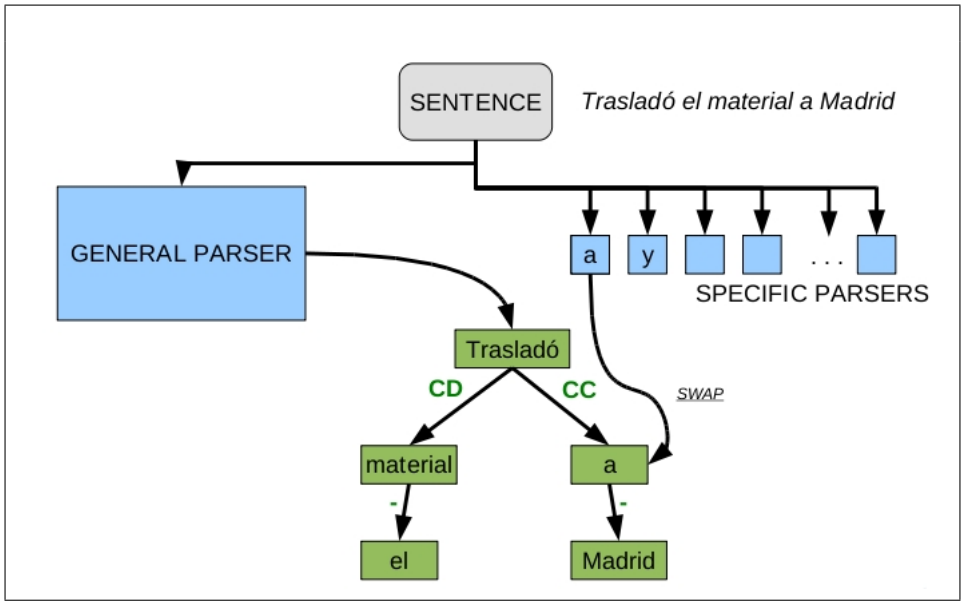

Figura 3.10: Mezcla de ambos árboles de dependencias obteniendo un árbol final con 100 % de precisión en el análisis para LAS, UAS y LA.

del desarrollo del analizador de n-versiones para la conjunción.

Primera Aproximación para el Estudio del Análisis de la Conjunción

Nuestra idea fue determinar si algunos tipos de frases 'difíciles' podían ser correctamente analizadas mediante la acción de unos analizadores específicos mientras el analizador general, entrenado con todo el corpus de entrenamiento, analizaría el resto. El primer analizador específico que tratamos de conseguir estaba entrenado para analizar frases que contenian una conjunción y que la frase completa estuviera entrecomillada. Esta situación se da numerosas veces en el corpus AnCora, y fue uno de los tipos de frases que marcamos como 'problemáticos'. De esta forma, entrenamos un analizador específico usando Maltparser 0.4 para frases que contenían conjunciones y estuvieran entrecomilladas. El analizador se entrenó utilizando la misma configuración que el grupo de Nivre en la CoNLL-X Shared Task para el castellano. El corpus de entrenamiento, obtenido automáticamente a partir del corpus de entrenamiento de AnCora, consistía en frases entrecomilladas y que contuvieran al menos una conjunción. Finalmente obtuvimos un corpus de 22 frases. El corpus de test se obtuvo de la misma forma, pero obteniendo las frases del corpus de test de AnCora que se usó en la CoNLL-X Shared Task. El corpus de test contenía 7 frases. Para analizar esta aproximación lo que hicimos fue construir incrementalmente el corpus de entrenamiento a

partir de las 22 frases, en primer lugar un corpus de una única frase, posteriormente un corpus con dos frases, y así sucesivamente. El método exacto que seguimos se explica a continuación:

- En primer lugar seleccionamos la frase con mayor número de palabras, ya que aporta mejor análisis si consideramos los resultados del experimento anterior, del subcorpus de entrenamiento de frases entrecomilladas y la añadimos como primera frase del corpus incremental.
- Después iteramos hasta que las 22 frases estuvieran incluidas en el corpus incremental. En cada iteración hicimos lo siguiente.
  - Se entrenó Malparser 0.4 con el corpus incremental.
  - Se analizó el corpus de test con el modelo entrenado.
  - La siguiente frase con mayor número de palabras se añadió al corpus incremental.

Los resultados de este experimento se muestran en la figura 3.11, donde se muestran los resultados de LAS, UAS y LA en cada iteración. En el eje de abscisas se representan el número de frases contenidas en el corpus incremental y en el eje de ordenadas los valores de LAS, UAS y LA.

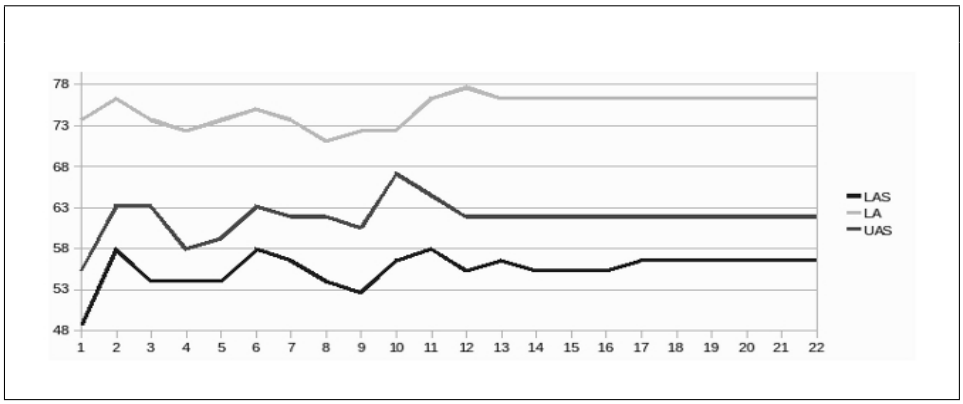

Figura 3.11: LAS, UAS y LA cuando se entrena un modelo específico, para frases entrecomilladas y que contengan conjunciones, incrementalmente usando el corpus AnCora.

Si sólo consideramos el análisis de la conjunción los resultados son muy buenos. En la primera iteración 3 conjunciones estaban mal analizadas, pero a partir de la segunda iteración sólo una conjunción estaba mal analizada. Pero si miramos los resultados de la Figura 3.11 los resultados globales no son tan buenos como los obtenidos por el analizador general. De todas formas, a pesar de que la precisión local para la conjunción se mejoró, esta aproximación no parece realista, ya que el número de ejemplos no es suficiente

para entrenar un modelo específico no sólo para obtener buenos resultados en el análisis de conjunciones también para todas las palabras que están en la frase entrecomillada. Este hecho nos hizo buscar otra aproximación que se explica en la siguiente sección.

Descripción de la Construcción del Analizador N-versiones para la Conjunción

En esta sección se muestra el estudio de una versión más compleja del analizador n-versiones. Como se puede ver en la subsección anterior, modelos específicos pueden ser entrenados para obtener alta precisión para el análisis de una palabra concreta, pero estos modelos no son capaces de analizar correctamente la frase completa donde la palabra concreta está contenida. Este es el hecho que inspiró la nueva aproximación: el analizador n-versiones como tal. La idea es obtener un número considerable de modelos específicos entrenados, cada uno capaz de analizar una palabra concreta en un patrón de comportamiento concreto, es decir, en un contexto específico. Por tanto la palabra concreta será una de las que son más frecuentemente mal analizadas y el contexto será uno de los patrones de etiquetados diferentes para esas palabras. Una de esas palabras es la conjunción y uno de los contextos en los que esta presente es el usado como ejemplo en la sección anterior, frases entrecomilladas. De esta forma, después de analizar una frase con el modelo general un programa debe decidir si la frase contiene una palabra que podría ser analizada por un analizador específico. En ese caso, el programa debe decidir que analizador específico es el más apropiado teniendo en cuenta el contexto en el que esta la palabra. Una vez que la frase sea analizada con el modelo específico, el resultado de la palabra "problemática" es reemplazado por el resultado obtenido en el modelo general. De esta forma, lo mejor de los dos analizadores puede ser aprovechado. En el caso de la conjunción, es algo más sencillo que en el resto de los casos ya que sólo hay que reemplazar de que palabra cuelga la conjunción ya que la relación de dependencia de la 'y' es siempre la misma: '-'. Para estudiar si este analizador n-versiones es útil para obtener mejores resultados realizamos el experimento que se describe a continuación.

Para el primer experimento se entrenó un modelo específico para las frases que contuvieran una conjunción actuando como coordinada copulativa. Se construyó un corpus de entrenamiento específico a partir del corpus de entrenamiento global, usando la sección de AnCora que fue propuesta para entrenamiento en la CoNLL-X Shared Task. Se obtuvo por tanto, un corpus de entrenamiento para frases en las que la conjunción actuaba de coordinada copulativa que finalmente contenía 361 frases (10.561 palabras o formas de palabra). Después se obtuvo un corpus de test por la misma via, usando el corpus de test que se propuso para el análisis en la CoNLL-X Shared Task. Por tanto, lo que se hizo fue analizar todas las frases que contenían con-

junciones actuando como coordinadas copulativas (16 frases, 549 palabras o formas de palabra).

Maltparser usa unos modelos de aprendizaje basados en historia, para predecir la siguiente acción que debe tomar en la derivación determinista del análisis de dependencias, lo que significa que usa características de la construcción del árbol de dependencias, en cuanto a la forma de construirlo, y características basadas en la forma de etiquetar los distintos nodos del árbol que dependen (o no) de otros. Más especificamente, las características en las que se basa el entrenamiento de los analizadores se definen en términos de la forma de la palabra (LEX), categorías gramaticales (POS), etiqueta de dependencia (DEP) de un elemento indicando sobre cual de las estructuras de datos, como se puede ver en la Sección 2.2.2. Usamos el modelo de características que Nivre propuso para el castellano en la CoNLL–X Shared Task, ver Figura 2.4, para llevar a cabo los experimentos descritos. Además para el presente experimento usamos el mismo modelo de características y obtuvimos que la conjunción se analizaba incorrectamente 8 veces (en un test con 16 conjunciones). Este hecho nos hizo investigar otros modelos diferentes para encontrar mejores resultados en los analizadores específicos. Después de una serie de malos modelos, encontramos un modelo capaz de analizar la conjunción correctamente en 12 de las 16 ocasiones, este modelo de características se observa en la Figura 3.12, y como se ve es un modelo más simple y concreto que el propuesto por Nivre.

```
POS     STACK
POS     INPUT
POS     INPUT   1
POS     INPUT   2
POS     INPUT   3
DEP     STACK
DEP     STACK   0       0       0       1
DEP     STACK   0       0       0       -1
DEP     INPUT   0       0       0       -1
FEATS   STACK
FEATS   INPUT
FEATS   INPUT   1
FEATS   STACK   0       0       1
LEX     STACK
LEX     INPUT
```

Figura 3.12: Modelo de características para las frases con conjunción actuando de coordinada copulativa.

Pero viendo los resultados obtenidos por el analizador general, que era capaz de analizar correctamente la conjunción en 13 de las 16 ocasiones, podría significar que los analizadores específicos no eran mejores que el analizador general y no eran viables para nuestros objetivos, pero viendo que los resultados de ambos analizadores eran muy parecidas, realizamos otra serie de experimentos para confirmar o rechazar nuestra hipótesis. Por tan-

to, entrenamos otra serie de analizadores para otros casos específicos de la conjunción.

El segundo experimento que llevamos a cabo fue el entrenamiento de un analizador específico para analizar conjunciones que actúan de nexo en una lista de nombres propios. Construimos un corpus de entrenamiento específico con las frases que contenían una sola conjunción actuando de nexo en una lista de nombres propios (al menos 2) usando para ello el corpus AnCora, más concretamente la sección usada para entrenamiento en la CoNLL–X Shared Task. El corpus resultante contenía 59 frases (1.741 palabras o elementos). Después de entrenar el analizador específico lo usamos para analizar todas las frases contenidas en el corpus de test que seguían el mismo patrón: el corpus de test resultante contenía tan solo 5 frases (121 palabras o elementos). El analizador resultante fue entrenado usando el mismo modelo de características que propuso Nivre en la CoNLL–X Shared Task. El analizador específico fue capaz de analizar correctamente las 5 conjunciones contenidas en el corpus de test, mientras que el analizador general entrenado con todo el corpus de test sólo fue capaz de analizar correctamente 4 de las 5 conjunciones.

Un tercer experimento se llevó a cabo para evaluar un modelo específico entrenado para analizar conjunciones que actúan como nexo en una lista de nombres comunes. Se construyó un corpus de entrenamiento específico con un conjunto de frases no ambiguas que contenían conjunciones actuando como conectora en una lista de nombres comunes, tomando las frases de la sección de AnCora que fue propuesta para entrenamiento en la CoNLL–X Shared Task. El corpus resultante contenía 266 frases (8.327 palabras o elementos). Después del entrenamiento se analizaron todas las frases que seguían ese mismo patrón seleccionándolas del corpus de test propuesto para análisis en la CoNLL–X Shared Task; el corpus resultante estaba formado por 15 frases (480 palabras o elementos). Una vez más el modelo de características propuesto por el grupo de Nivre fue el mejor. El modelo específico fue capaz de analizar correctamente 12 de las 15 conjunciones, mientras que el modelo general, entrenado con el corpus de entrenamiento completo, tan sólo analizó correctamente 10 de las 15 conjunciones.

Un último experimento se realizó para descubrir como de viables son los analizadores n–versiones. De esta manera, entrenamos un modelo específico para analizar conjunciones actuando como conectiva en una lista de adjetivos o construcciones con valor de adjetivo. Construimos un corpus de entrenamiento específico con el conjunto de frases no ambiguas que contenían conjunciones que conectaban adjetivos o construcciones actuando como adjetivos, de la sección de AnCora que se propuso para entrenamiento en la CoNLL–X Shared Task. Este corpus específico contenía 59 frases (3.155 palabras o elementos). Después de entrenar el analizador específico seleccionamos del corpus de test las frases que seguían el mismo patrón de la sección de AnCora propuesta para análisis en la CoNLL–X Shared Task.

El corpus de test resultante contenía 5 frases (113 palabras o elementos). El modelo de características del grupo de Nivre volvió a dar el mejor resultado de nuevo, ya que el analizador específico, entrenado bajo esas características, fue capaz de analizar correctamente las 5 conjunciones contenidas en esas frases, mientras que el modelo general sólo analizó correctamente 4 de las 5.

La tabla 3.6 muestra los resultados obtenidos por los analizadores específicos de manera general para la conjunción.

Tabla 3.6: Estudio del análisis de la conjunción. La flecha hacia la izquierda ($^{\leftarrow}$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3	#4
Etiqueta	–	–	–	–
Conectado con	verbo$^{\leftarrow}$	nombre propio$^{\leftarrow}$	nombre común$^{\leftarrow}$	adjetivo$^{\leftarrow}$
LAS$_{y/e}$ antes	***81,3*** %	80 %	66,7 %	80 %
LAS$_{y/e}$ después	75 %	***100*** %	***80*** %	***100*** %
# Frases corpus test	16	5	15	5
# Frases corpus train	361	59	266	59

Los análisis obtenidos por el modelo general para las conjunciones que seleccionamos para este experimento, se reemplazaron por los resultados obtenidos por los analizadores específicos, obteniendo así unos resultados globales de LAS = 81.92 %, UAS = 85.31 % and LA = 90.06 %, la mejora no es muy grande pero es importante decir, que las conjunciones dejaron de estar dentro del conjunto de palabras que devolvían un peor análisis. Estos resultados nos animaron a realizar experimentos similares con el resto de palabras críticas.

3.2.4. La Preposición 'a'

En esta sección se expone como se realizó el estudio de viabilidad para el analizador de N–versiones para el caso de la preposición 'a'. La preposición 'a' (como el resto de las preposiciones presentes en el corpus AnCora), a diferencia de la conjunción, puede etiquetarse de diversas maneras, lo que en un principio suponía un nuevo problema. Se procedió de la siguiente manera: continuando con el experimento obtenido para la conjunción, se fueron obteniendo las diferentes formas de etiquetar la preposición 'a' y para cada caso se generó mediante la programación de unos scripts, corpora específico para el entrenamiento y análisis de cada caso.

La preposición 'a', al igual que todas las preposiciones, tiene una gran importancia en el análisis ofrecido por los creadores de AnCora, ya que utilizan las preposiciones como raíces de los subárboles que amplían la información de la acción principal de la frase. Si consideramos la frase *Trasladó el mate-*

rial a Madrid, no es *Madrid* la raíz del subárbol, sino que lo es la preposición 'a', por lo que al producir un error en el análisis de la misma, se está provocando un error en toda la rama. El ejemplo, es un caso muy sencillo, pero pueden existir ramas de gran longitud y para algunos propósitos el error provodado por las preposiciones puede provocar la inutilidad de una gran parte de un árbol de dependencias.

Esta palabra es la preposición que obtiene un peor análisis por el analizador general por lo que interesaba estudiarla en primer lugar y comprobar que los buenos resultados obtenidos en el estudio de la conjunción se repetían.

Se obtuvieron 7 casos de diferente etiquetado de la 'a', que involucraban a 29 frases en el corpus de test; se consiguió mediante la replicación de las técnicas presentadas en la Sección 3.2.3, pasar de 15 etiquetados erróneos a tan sólo 3, es decir, la precisión LAS pasó del 48,28 % al 93,10 % en este corpus de test para el análisis de la preposición 'a'.

La 'a' puede etiquetarse a un verbo como CD (Complemento Directo), CI (Complemento Indirecto), CC (Adjunto), CREG (Complemento Preposicional) o como – (etiqueta nula), y a un nombre como – (etiqueta nula). Para todos los casos se obtuvieron mejores resultados con el analizador específico que con el general.

En la Tabla 3.7 se muestran los resultados para la preposición 'a' que a continuación se exponen.

Tabla 3.7: Estudio del análisis de la preposición 'a'. La flecha hacia la izquierda ($\leftarrow$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($\rightarrow$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3	#4	#5	#6
Etiqueta	CD	CI	CC	CREG	–	–
Conectado con	verbo$^{\leftarrow}$					nombre$^{\leftarrow}$
LAS$_{y/e}$ antes	62,5 %	42,9 %	60 %	25 %	0 %	50 %
LAS$_{y/e}$ después	***87,5*** %	***100*** %	***100*** %	***75*** %	0 %	***100*** %
# Frases corpus test	8	7	5	4	1	4
# Frases corpus train	110	80	146	86	8	63

En primer lugar se generó un corpus específico que contenía sólo frases con una aparición de la preposición 'a' etiquetada como CD de la sección del corpus de AnCora propuesto para entrenamiento en la CoNLL–X Shared Task; el corpus resultante contenía 110 frases. Se hizo lo mismo con el corpus de test, obteniendo un corpus con 8 frases. El analizador general analizó correctamente 5 de las 8 preposiciones, mientras que nuestro modelo específico mejoró esos datos analizando correctamente 6 de las 8.

Se continuó con el experimento con el siguiente caso: se seleccionó de los corpora de entrenamiento y análisis las frases que contenían una sóla 'a'

etiquetada como CI y conectada a un verbo; se obtuvo un corpus de entrenamiento de 80 frases, y un corpus de test de 7 frases. El analizador general fue capaz de analizar correctamente sólo 3 de las 7 frases, mientras que nuestro analizador específico analizó correctamente las 7 frases, obteniendo en este caso un 100 % LAS para la preposición 'a'.

Para el caso en el que la preposición 'a' se etiqueta mediante CC y se conecta a un verbo se procedió de igual manera, obteniendo un corpus de entrenamiento de 146 frases y un corpus de test de 5 frases. El analizador general falló en 2 de las 5 ocasiones mientras que el analizador específico fue capaz de analizar perfectamente las 5 preposiciones.

Se procedió de igual manera para el caso en el que la preposición 'a' estaba etiquetada como CREG y conectada a un verbo, obteniendo un corpus de entrenamiento con 86 frases y un corpus de test de 4 frases; el modelo general produjo un análisis de tan sólo 25 % LAS para la 'a', mientras que el modelo específico fue capaz de obtener un 75 % LAS, es decir analizó correctamente 3 de las 4 preposiciones.

El caso en el que la preposición 'a' se conecta a un verbo mediante la etiqueta –, es un caso particular en el que el corpus de test tan sólo tiene una frase, y el de entrenamiento 8 frases. El analizador general no etiqueta correctamente la frase pero realiza correctamente la conexión, mientras que el analizador específico hace justo lo contrario. Es interesante quizá enriquecer estos casos de los corpora para así obtener un mejor análisis global ya que son casos poco representativos y frecuentes.

Por último, tenemos el caso en el que la preposición 'a' está conectada a un nombre (común o propio) mediante la etiqueta –. Para este caso existen 63 frases en el corpus de entrenamiento y 4 frases en el corpus de test. El analizador general analizó correctamente 2 de las 4 frases, mientras que el analizador específico lo hizo para las 4 frases.

Los resultados globales obtenidos tras las mejoras por el nuevo análisis de la preposición 'a' fueron los siguientes, LAS = 82,17 %, UAS = 85,51 %, LA = 90,32 %. Como se puede observar, los resultados de la preposición 'a' fueron muy buenos, por tanto eso animó a continuar con el resto de palabras que estaban mal analizadas por el analizador general y de esa manera obtener un mejor análisis.

3.2.5. La Preposición 'de'

El procedimiento seguido para obtener un mejor análisis para la preposición 'de' fue muy similar al de la preposición 'a'. Se encontraron 5 patrones diferentes seguidos por esta palabra en el corpus AnCora. Preposición 'de' conectada a nombre con etiqueta –, preposición 'de' conectada a adverbio o adjetivo con etiqueta –, preposición 'de' conectada a verbo, la etiqueta es CC, preposición 'de' conectada a verbo con etiqueta CREG, y por último, preposición 'de' conectada a verbo cuyo identificador es mayor que el de la

preposición.

En la Tabla 3.8 se muestran los resultados obtenidos por los analizadores específicos para la preposición 'de'.

Tabla 3.8: Estudio del análisis de la preposición 'de'. La flecha hacia la izquierda ($^{\leftarrow}$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3	#4
Etiqueta	CC	CREG	–	–
Conectado con	verbo$^{\leftarrow}$		adverbio o adjetivo$^{\leftarrow}$	nombre$^{\leftarrow}$
LAS$_{y/e}$ antes	0 %	0 %	***100*** %	83,3 %
LAS$_{y/e}$ después	***100*** %	***100*** %	***100*** %	***96,7*** %
# Frases corpus test	30	5	2	2
# Frases corpus train	535	105	39	32

Para el primer caso, en el que la preposición 'de' está conectada a nombres (comunes y propios) y la etiqueta es – (etiqueta nula), se generaron corpora de entrenamiento y test de manera automática mediante la programación de un script en Perl, obteniendo un corpus de entrenamiento de considerable tamaño, 535 frases, y un corpus de test de 30 frases. El analizador general producía fallo en 5 de las 30 frases, mientras que el analizador específico tan sólo lo hacía 1 vez.

En el segundo caso, cuando la preposición 'de' está conectada a adjetivos o adverbios (y formaciones con forma de adjetivo o adverbio), se obtuvo un corpus de entrenamiento que contaba con 105 frases y un corpus de test de 5 frases. Tanto el analizador general como el específico fueron capaces de analizar perfectamente los 5 casos.

Seguidamente se continuó con el experimento, buscando de forma automática en los corpora el caso en el que la preposición 'de' estaba conectada a un verbo cuando la etiqueta es CC. El corpus de entrenamiento generado contenía 39 frases y el corpus de test tan sólo 2 frases. El analizador general produjo fallo en ambos casos, mientras que el específicos los analizó bien.

De la misma manera se estudió el caso en el que la preposición 'de' estaba conectada a verbo y la etiqueta era CREG; se obtuvo un corpus de entrenamiento que contenía 32 frases y un corpus de test de 2 frases. El analizador general produjo fallo en ambos casos, mientras que el analizador específico fue capaz de analizarlos perfectamente.

Por último se estudió un caso que tuvo una particularidad especial, ya que no se pudieron obtener datos reales porque el corpus de test no contenía ejemplos de ese patrón. Se trataba de la preposición 'de' conectada a verbos cuyo identificador fuera mayor que el identificador de la preposición lo que quiere decir que está escrito después en la frase. El corpus de entrenamiento obtenido tenía 12 frases, mientras que el corpus de test contenía 0 frases.

Estos resultados nos llevaron a pensar en que a la hora de producir corpora para el análisis de dependencias es más inteligente realizarlo de manera equitativa y que aparezcan todo tipo de frases en ambos corpora.

Los resultados globales obtenidos tras el análisis específico de la preposición 'de' fueron los siguientes,LAS = 82,02 %, UAS = 85,27 % y LA = 90,44 %. Lo que mantiene la viabilidad de la propuesta ya que estamos produciendo mejora en todos los casos.

3.2.6. El Nexo 'que'

El nexo 'que' tiene menos variedad a la hora de analizarlo mediante análisis de dependencias. Sólo sigue 3 patrones diferentes. Los patrones que sigue son los siguientes: nexo 'que' conectado a verbo, cuyo identificador es menor que el del nexo, como SUJ, nexo 'que' conectado a verbo, cuyo identificador es menor que el del nexo, mediante la etiqueta –, y finalmente nexo 'que' conectado a verbo cuyo identificador es mayor que el del nexo.

Esta palabra tiene un comportamiento diferente al resto de las palabras estudiadas, ya que es más probable que esté conectada a palabras que están escritas después en la frase. Por ello, el estudio de distintos modelos de características era obligado.

En la Tabla 3.9 se exponen los resultados del experimento del nexo 'que' de manera general, que se exponen a continuación.

Tabla 3.9: Estudio del análisis del nexo 'que'. La flecha hacia la izquierda ($\leftarrow$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($\rightarrow$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3
Etiqueta	SUJ	–	SUJ
Conectado con	verbo$\rightarrow$		verbo$\leftarrow$
LAS$_{y/e}$ antes	82,5 %	86,4 %	0 %
LAS$_{y/e}$ después	***92,3*** %	***95,5*** %	***100*** %
# Frases corpus test	26	22	1
# Frases corpus train	349	342	6

El primer caso de estudio fue en el que el nexo 'que' estaba conectado a un verbo cuyo índice era mayor que el índice del nexo. Para el entrenamiento del analizador específico se utilizó el modelo de características de la Figura 3.13. El corpus de entrenamiento obtenido contenía 349 frases y el de test 26 frases. El modelo general, entrenado mediante el mismo modelo presentado por el grupo de Nivre en la CoNLL–X Shared Task, analizó incorrectamente 3 de las 26 frases, mientras que el analizador específico entrenado mediante el modelo de características de la Figura 3.13, analizó incorrectamente 2 de las 26 frases.

```
POS     STACK
POS     INPUT
POS     INPUT   1
DEP     STACK
DEP     STACK   0       0       0       1
DEP     STACK   0       0       0       -1
DEP     INPUT   0       0       0       -1
```

Figura 3.13: Modelo de características utilizado para el caso en el que el nexo se conecta a un verbo como SUJ.

Se continuó con el estudió del nexo 'que', observando el caso en el que el nexo se conecta a un verbo cuyo identificador es mayor que el del propio nexo y la etiqueta es –. El corpus de entrenamiento obtenido contenía 342 frases, y el de test contenía 22 frases. El analizador general produjo 3 errores y el analizador específico, entrenado mediante el modelo de características de la Figura 3.14, ya que el modelo propuesto por Nivre producía mayor cantidad de fallos, produjo tan sólo 1 fallo.

```
POS     STACK
POS     INPUT
POS     INPUT   1
POS     INPUT   2
POS     INPUT   3
POS     STACK   1
POS     STACK   0       0       1
POS     STACK   0       0       0       -1
POS     STACK   0       0       0       1
POS     INPUT   0       0       0       -1
POS     STACK   0       1
POS     INPUT   0       -1
POS     STACK   2
FEATS   STACK
FEATS   INPUT
FEATS   INPUT   1
```

Figura 3.14: Modelo de características utilizado para el caso en el que el nexo se conecta a un verbo como –.

Por último se estudió el caso en el que el nexo 'que' estaba conectado a un verbo cuyo identificador es menor que el del verbo y la etiqueta es SUJ. En este caso el corpus de entrenamiento contenía tan solo 6 ejemplos y el corpus de test 1 ejemplo. El analizador general no fue capaz de analizar correctamente la frase, mientras que el analizador específico, en este caso si, entrenado mediante el mismo modelo específico que el general, fue capaz de hacerlo.

Los resultados globales obtenidos tras el experimento del nexo 'que' llevaron a obtener los siguientes resultados, LAS = 81,94 %, UAS = 85,27 % y LA = 90,36 %. Obteniendo, de esta manera, mejores resultados que el modelo general.

3.2.7. La Preposición 'en'

Para el estudio de viabilidad para la mejora del análisis de la preposición 'en' se procedió de manera similar que para el resto de palabras, salvando las diferencias entre unas y otras. Se detectaron 4 patrones diferentes que la preposición 'en': 'en' conectado a verbo cuyo identificador es mayor que el de la preposición con la etiqueta CC, 'en' conectado a verbo cuyo identificador es menor que el de la preposición con la etiqueta CC, 'en' conectado a verbo con la etiqueta CREG y cuyo identificador es menor que el de la preposición 'en' y finalmente 'en' conectado a nombre con etiqueta –.

En la Tabla 3.10 se exponen los resultados obtenidos por los analizadores específicos para la preposición 'en' de manera general, que a continuación se exponen.

Tabla 3.10: Estudio del análisis de la preposición 'en'. La flecha hacia la izquierda ($^{\leftarrow}$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3	#4
Etiqueta	CC	CC	CREG	–
Conectado con	verbo$^{\rightarrow}$	verbo$^{\leftarrow}$		nombre$^{\leftarrow}$
LAS$_{y/e}$ antes	***83,3*** %	92,6 %	50 %	62,5 %
LAS$_{y/e}$ después	***83,3*** %	***100*** %	***100*** %	***87,7*** %
# Frases corpus test	6	27	2	8
# Frases corpus train	111	363	55	121

Para el primer caso, se generaron corpora de entrenamiento y análisis que contenía 111 y 6 frases respectivamente. Tanto el analizador general como el analizador específico fueron capaces de analizar correctamente 5 de las 6 frases. Provocando ambas un fallo en la misma frase.

En el segundo caso, 'en' conectado a verbo cuyo identificador es menor que el de la preposición con la etiqueta CC, se obtuvo un corpus de entrenamiento de 363 frases y un corpus de test de 27 frases. El modelo específico obtuvo LAS = 100 % para el análisis de la preposición mientras que el modelo general produjo 2 fallos obteniendo un LAS = 92,6 %.

Seguidamente se estudió el caso en el que la preposición 'en' estaba conectada a verbo mediante CREG. Se obtuvo un corpus de entrenamiento de 55 frases y un corpus de test de 2 frases. El analizador general produjo un fallo, mientras que el modelo específico produjo 0 fallos.

Por último se estudió el caso de la preposición 'en' conectada a nombre con la etiqueta –. Se obtuvo un corpus de entrenamiento de 121 frases y un corpus de test de 8 frases. El analizador específico provocó un fallo en las 8 preposiciones que analizó, mientras que el modelo general produjo 3 fallos.

Los resultados globales obtenidos tras el estudio y mejora del análisis de

la preposición 'en' fueron, LAS = 81,96 %, UAS = 85,31 %, y LA = 90,36 %. Obteniendo, una vez más, un análisis de mayor calidad.

3.2.8. La Preposición 'con'

Para la preposición 'con' se procedió de la misma manera, encontrando para este caso 4 patrones diferentes a la hora de analizar esta preposición. Puede conectarse a un verbo como CC, CREG o –, y puede conectarse a un nombre como –.

En la Tabla 3.11 se muestran de manera general los resultados obtenidos por los modelos específicos que se exponen a continuación.

Tabla 3.11: Estudio del análisis de la preposición 'con'. La flecha hacia la izquierda ($^{\leftarrow}$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3	#4
Etiqueta	CC	CREG	–	–
Conectado con	verbo$^{\leftarrow}$			nombre$^{\leftarrow}$
LAS$_{y/e}$ antes	60 %	40 %	***100*** %	66,7 %
LAS$_{y/e}$ después	***80*** %	***100*** %	***100*** %	***83,3*** %
# Frases corpus test	10	5	1	6
# Frases corpus train	204	39	5	95

Para el primer caso en el que la preposición 'con' se conecta a un verbo y la etiqueta es 'CC', se obtuvo un corpus de entrenamiento de 204 frases y un corpus de test de 10 frases. El analizador general fue capaz de analizar correctamente 6 de las 10 frases, mientras que el modelo específico lo hizo en 8 de las 10 frases.

Siguiendo con el estudio de la preposición 'con' se observó el caso en el que se conecta a un verbo con la etiqueta CREG, obteniendo un corpus de entrenamiento de 39 frases y un corpus de análisis de 5 frases. El modelo general provocaba 3 fallos, mientras que el modelo específico ninguno.

El caso en el que la preposición 'con' se conecta a un verbo mediante – es un ejemplo que tiene muy pocas apariciones, tan sólo 5 frases en el corpus de entrenamiento y 1 frase en el de test. Ambos analizadores, específico y general, fueron capaces de analizar correctamente la preposición contenida en el ejemplo.

Por último, se estudió el caso en el que la preposición 'con' se conectaba a un nombre con la etiqueta –. Existen 95 frases en el corpus de entrenamiento y 6 frases en el de test. El analizador general produjo 2 fallos, mientras que el analizador específico produjo sólo 1.

Los resultados globales obtenidos tras el estudio de la preposición 'con' fueron los siguientes: LAS = 81,96 %, UAS = 85,31 % y LA = 90,40 %.

3.2.9. La Preposición 'por'

La preposición 'por' puede conectarse a un nombre, una coma, o un adjetivo y puede estar etiquetada como – o como CAG. Por tanto, se tienen 3 casos diferentes: 'por' conectada a nombre con –, 'por' conectada a coma como CAG, y 'por' conectada a un adjetivo como CAG.

En la Tabla 3.12 se muestran los resultados de los analizadores específicos para la preposición 'por' que se exponen a continuación.

Tabla 3.12: Estudio del análisis de la preposición 'por'. La flecha hacia la izquierda ($^{\leftarrow}$) indica que la conexión es hacía una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

Caso	#1	#2	#3
Etiqueta	–	CAG	CAG
Conectado con	nombre$^{\leftarrow}$	coma$^{\leftarrow}$	adjetivo$^{\leftarrow}$
LAS$_{y/e}$ antes	***100*** %	***100*** %	80 %
LAS$_{y/e}$ después	***100*** %	***100*** %	***100*** %
# Frases corpus test	1	1	5
# Frases corpus train	47	13	71

Para el primer caso se obtuvo un corpus de entrenamiento de 47 frases y un corpus de test de 1 frase. Tanto el analizador específico como el general fueron capaces de analizar perfectamente la preposición 'por' de la frase del corpus de test.

En el segundo caso ocurre lo mismo, pero teniendo 13 frases en el corpus de entrenamiento y 1 frase en el corpus de test.

Finalmente, para el caso en el que la preposición 'por' está conectada a un adjetivo mediante la etiqueta CAG, se obtienen 71 frases en el corpus de entrenamiento y 5 frases en el corpus de test, siendo el analizador general capaz de analizar correctamente 4 de las 5 frases, y el analizador específico las 5 frases.

Los resultados globales obtenidos tras el estudio de la preposición 'por' fueron los siguientes: LAS = 81,96 %, UAS = 85,31 %, y LA = 90,36 %.

3.2.10. Resultados Obtenidos por el Analizador N–Versiones

En esta sección se presentan los resultados obtenidos al desarrollar un experimento parecido para cada una de las palabras que son peor analizadas por un analizador general entrenado con el mismo corpus de entrenamiento y mismo modelo de características, propuesto por Nivre, que en la CoNLL–X Shared Task para el castellano. Se compara el resultado del analizador general con el modelo específico para cada caso, expuestos en las secciones anteriores.

Tabla 3.13: Etiquetado de todas las palabras estudiadas de AnCora. En cada caso se muestra el resultado (LAS) antes y después de aplicar nuestro método. La flecha hacia la izquierda ($^{\leftarrow}$) indica que esa conexión es hacia una palabra que está situada delante en la frase. La flecha hacia la derecha ($^{\rightarrow}$) indica que la palabra a la que se conecta está situada después en la frase.

		Caso					
Palabra		#1	#2	#3	#4	#5	#6
y/e	**Etiqueta**	–	–	–	–		
	Conectado con	verbo$^{\leftarrow}$	nombre propio$^{\leftarrow}$	nombre común$^{\leftarrow}$	adjetivo$^{\leftarrow}$		
	LAS$_{y/e}$ antes	***81,3*** %	80 %	66,7 %	80 %		
	LAS$_{y/e}$ después	75 %	***100*** %	***80*** %	***100*** %		
a	**Etiqueta**	CD	CI	CC	CREG	–	–
	Conectado con	verbo$^{\leftarrow}$					nombre$^{\leftarrow}$
	LAS$_a$ antes	62,5 %	42,9 %	60 %	25 %	***0*** %	50 %
	LAS$_a$ después	***87,5*** %	***100*** %	***100*** %	***75*** %	***0*** %	***100*** %
de	**Etiqueta**	CC	CREG	–	–		
	Conectado con	verbo$^{\leftarrow}$		adverbio$^{\leftarrow}$ adjetivo$^{\leftarrow}$	nombre$^{\leftarrow}$		
	LAS$_{de}$ antes	0 %	0 %	***100*** %	83,3 %		
	LAS$_{de}$ después	***100*** %	***100*** %	***100*** %	***96,7*** %		
que	**Etiqueta**	SUJ	–	SUJ			
	Conectado con	verbo$^{\rightarrow}$		verbo$^{\leftarrow}$			
	LAS$_{que}$ antes	88,5 %	86,4 %	0 %			
	LAS$_{que}$ después	***92,3*** %	***95,5*** %	***100*** %			
en	**Etiqueta**	CC	CC	CREG	–		
	Conectado con	verbo$^{\rightarrow}$	verbo$^{\leftarrow}$		nombre$^{\leftarrow}$		
	LAS$_{en}$ antes	***83,3*** %	92,6 %	50 %	62,5 %		
	LAS$_{en}$ después	***83,3*** %	***100*** %	***100*** %	***87,5*** %		
con	**Etiqueta**	CC	CREG	–	–		
	Conectado con	verbo$^{\leftarrow}$			nombre$^{\leftarrow}$		
	LAS$_{con}$ antes	60 %	40 %	***100*** %	66,7 %		
	LAS$_{con}$ después	***80*** %	***100*** %	***100*** %	***83,3*** %		
por	**Label**	–	CAG	CAG			
	Conectado con	nombre$^{\leftarrow}$	coma$^{\leftarrow}$	adjetivo$^{\leftarrow}$			
	LAS$_{por}$ antes	***100*** %	***100*** %	80 %			
	LAS$_{por}$ después	***100*** %	***100*** %	***100*** %			

Como se puede observar los resultados obtenidos indican una clara mejora en el análisis de esas palabras *difíciles* de analizar ya que hay mejora en la mayoría de los patrones y sólo hay un caso en el que el analizador general es mejor que el específico.

En algunos casos la mejora parece espectacular. Por ejemplo, si observamos el análisis de la preposición *de* cuando se conecta a un verbo, el analizador general no es capaz de analizar correctamente ninguna, por tanto su resultado es un 0 % de LAS y el analizador específico es capaz de analizar correctamente todas. Pero cabe decir que es debido al reducido número de ejemplos en el corpus de test para este patrón concreto, de hecho si sólo hay una frase en el corpus de test y un analizador es capaz de analizarla correctamente, el resultado será 100 %, por tanto hay que tener en cuenta este detalle a la hora de evaluar el resultado. Aunque para el caso de estudio,

la preposición *de* conectada a verbos, el corpus de test está formado por 4 frases y todos los ejemplos estaban mal analizados por el analizador general y sin embargo el analizador específico fue capaz de analizar correctamente los 4. Si el corpus de test contiene hubiera contenido más ejemplos quizá los resultados devueltos no habrían sido tan radicales. La mayoría de los corpus de test específicos contienen entre 2 y 10 frases, pero hay algunos con más de 30. Como se dice en (Ballesteros et al., 2010a), la precisión en el análisis es razonablemente homogénea y se espera obtener los mismos resultados cuando se aumenta el número de ejemplos en el corpus de test.

Además hay que decir que el caso de la preposición *de* conectada a un verbo con la etiqueta "–" es un caso que está contenido numerosas veces en el corpus de entrenamiento, pero que no está presente en el corpus de test. Por supuesto, para esta situación no hay error producido por el analizador general. Por tanto, quizá una buena sugerencia de trabajo futuro es encontrar otras fórmulas a la hora de construir corpora de entrenamiento y test, y así obtenter mejor precisión en el análisis.

3.2.11. Precisión Global, Precisión Local y sus Límites

Como se puede ver en (Ballesteros et al., 2010a) y (Ballesteros et al., 2010c) y en la Subsección 3.2.10, como resultado de usar analizadores específicos la precisión local se mejora. Por tanto, también se obtiene mejor precisión global. Los analizadores de dependencias pueden ser útiles para múltiples tareas de tratamiento de lenguaje natural, como la búsqueda de la parte negada de una frase, simplificación de textos y muchos otros propósitos, es importante que los árboles analizados de dependencias tengan una precisión elevada para que puedan ser utilizados correctamente. Por tanto, ni siquiera un error debe ser admisible para el usuario final. Después de analizar el corpus de test con nuestro analizador de n–versiones observamos que 42 (20,3 %) de las frases no mostraban ningún error, mientras que 38 (18,4 %) lo hacían tras analizarlas con el analizador general. Esta mejora de la precisión local como se puede leer en (Ballesteros et al., 2010a) tiene una importante consecuencia, el usuario final quedará más satisfecho.

Combinando los resultados del analizador general con los analizadores específicos como se describe en la subsección 3.2.2, obtuvimos los siguientes resultados: LAS = 82,68 %, UAS = 85,73 % and LA = 90,84 %. Lo que es un incremento de 1,38 % LAS, 1,06 % UAS y 0,78 % LA globales con respecto a nuestros resultados de partida: los del analizador general. Estos resultados pueden verse de manera extendida en las Figuras 3.15, 3.16, 3.17 y 3.18.

Los analizadores N–versiones son una forma de mejorar la precisión en el análisis sistemáticamente evitando los errores dados por el analizador general. Aún así nuestros experimentos aportan una mejora discreta. Esta mejora es más relevante cuando se eliminan errores causados por una palabra frecuente, como se muestra en las Figuras 3.15, 3.16 y 3.17. En cada figura cada

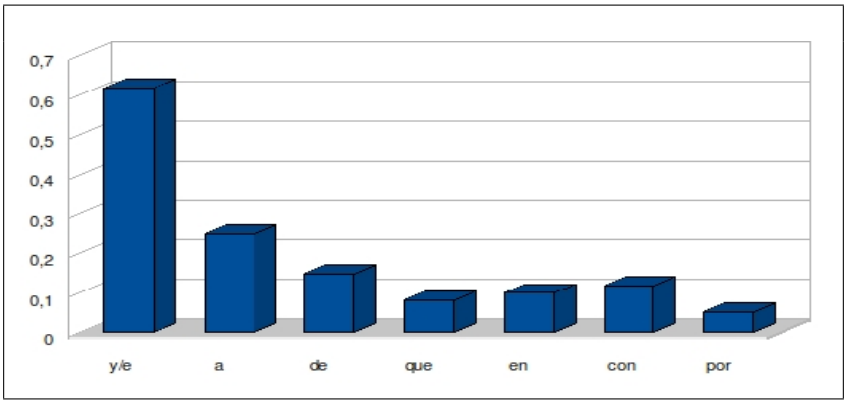

Figura 3.15: Incremento de LAS global debido a la mejora obtenida usando cada analizador específico.

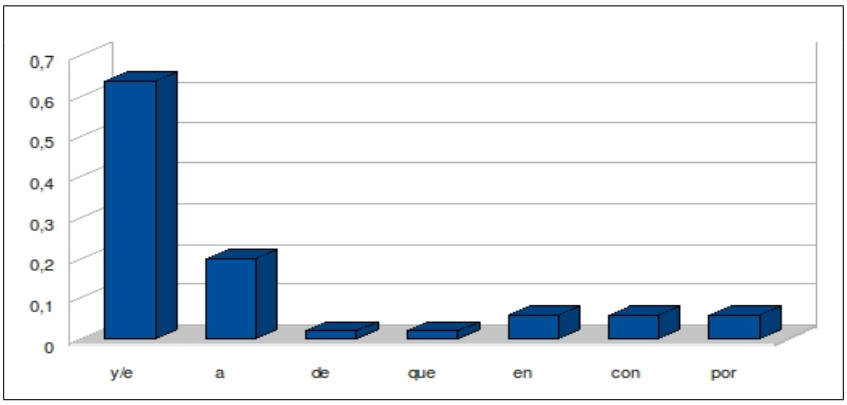

Figura 3.16: Incremento de UAS global debido a la mejora obtenida usando cada analizador específico.

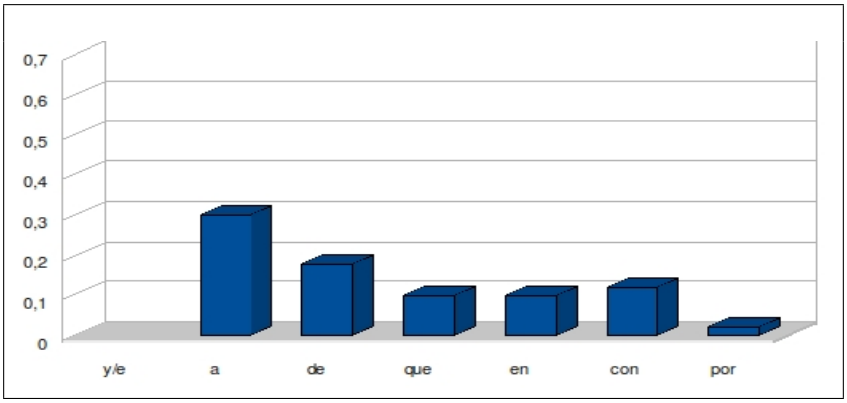

Figura 3.17: Incremento de LA global debido a la mejora obtenida usando cada analizador específico.

conjunto de barras muestra el incremento de LAS, UAS y LA que aporta cada palabra. La primera palabra para la que hicimos el experimento fue la conjunción (*y* o *e*). Esto es porque la conjunción es la palabra que tiene un mayor número de errores, es decir, es la más frecuentemente peor analizada por el analizador general. Siguiendo con esta idea, se añadieron acumulando las mejoras producidas por el resto de analizadores, seleccionando en primer lugar aquellas palabras que tienen un mayor número de errores. Al final, cuando se acumula la acción de todos los analizadores, tenemos el resultado global completo mostrado en la Tabla 3.13. Se puede observar en las Figuras 3.15, 3.16 y 3.17 que LAS, UAS y LA se incrementan notablemente con la acción de los analizadores específicos de la conjunción y de la preposición *a*. De hecho, LA no obtiene mejora con la acción del analizador específico para la conjunción ya que el analizador general no comete errores cuando etiqueta la relación de dependencia de la conjunción pero comete muchos a la hora de decidir con que otra palabra se relaciona. En cualquier caso, en términos generales hay que focalizar los resultados en la obtención de una mejor precisión local, no tanto como una mejor precisión global. Es interesante recalcar que la conjunción provoca 56 errores tras el análisis, *'a'* provoca 48 errores, *'de'* 44 errores, *'que'* 42 errores, *'en'* 37 errores, *'con'* 17 errores y *'por'* 16 errores. Además, los incrementos obtenidos y la precisión no son regulares por el número de ejemplos de cada caso considerado.

3.2.12. Conclusiones del Experimento

Como se puede ver en la Figura 3.18 se está consiguiendo producir una mayor precisión en el análisis debido a la acción de los analizadores específicos. De manera que se puede deducir lo que es una sugerencia obligada de trabajo futuro: seguir buscando palabras que produzcan mayor cantidad de errores, haciendo cada vez más complejo el analizador final y conseguir de esa manera un análisis más completo.

Además, aunque un gran porcentaje de los errores más comunes se evitan usando el analizador n–versiones, aún queda una cantidad considerable de nodos mal etiquetados producidos por el analizador general. Después de nuestros esfuerzos un 17,32 % de mejora en LAS sigue siendo necesaria para obtener un análisis perfecto. Pero cabe destacar que las palabras que se han tratado en este estudio han dejado de ser las que peor calidad de análisis mostraban. Es importante tener en cuenta que estas palabras (conjunción, preposiciones, etc.) tienen una relevancia mayor que el resto, ya que suelen ser raíz de importantes partes del árbol, como es el caso de las preposiciones. Como se puede ver en las imágenes de la Figura 3.19 el error producido tras el análisis de la preposición *con* produce que toda una rama del árbol esté incorrectamente analizada, a pesar de que el resto de elementos pertenecientes a esa rama estén correctamente analizados por el analizador general. Por tanto, se concluye, que la mejora en el análisis de

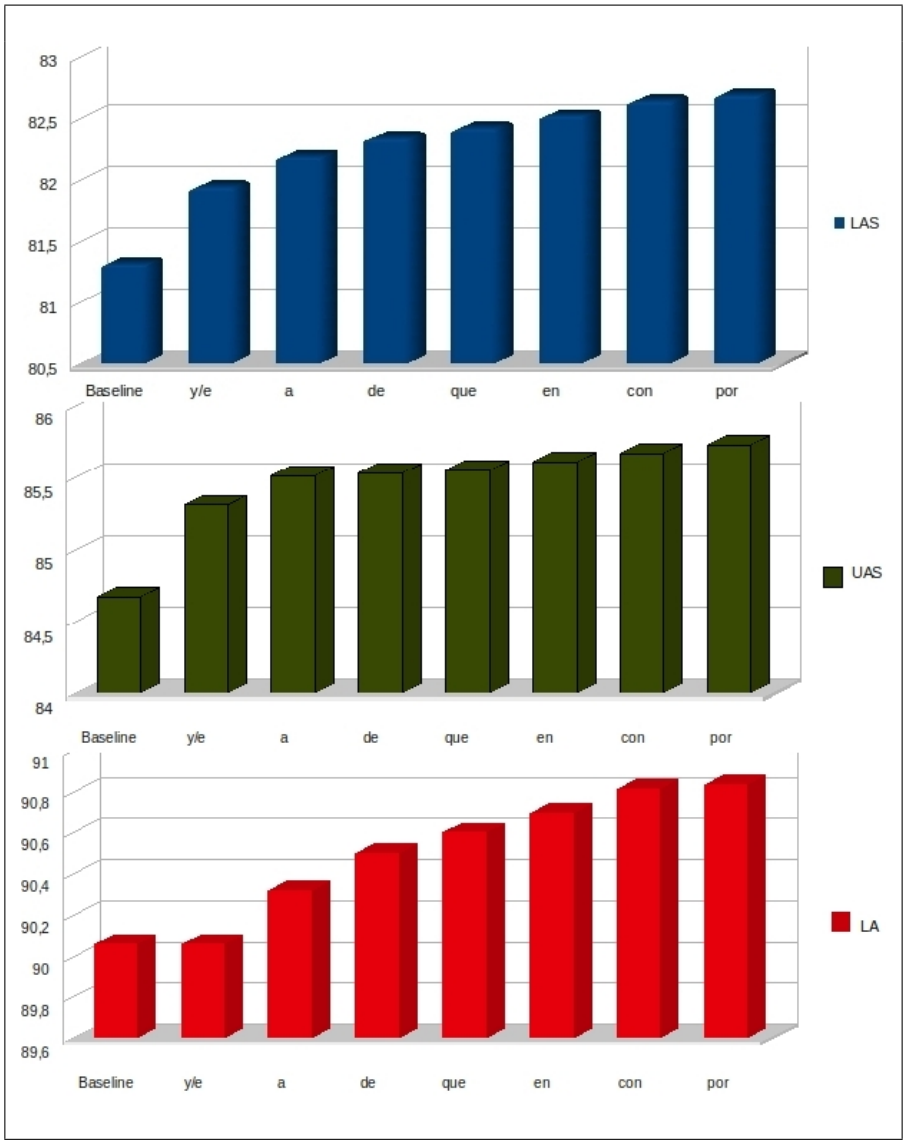

Figura 3.18: Mejora global obtenida para cada una de las 3 medidas de evaluación.

este conjunto de palabras *problemáticas* genera un análisis de dependencias de mucha mayor calidad.

Este fenómeno, por el cual un árbol resultante analizado está mal construido debido a un análisis incorrecto de una de las palabras analizadas en este estudio, se repite constantemente en el corpus de test. Por tanto, de esta manera no sólo estamos mejorando la precisión global ya que estamos produciendo un mejor análisis en los nodos que contienen esas palabras, sino que además y lo que en definitiva es más importante, estamos produciendo

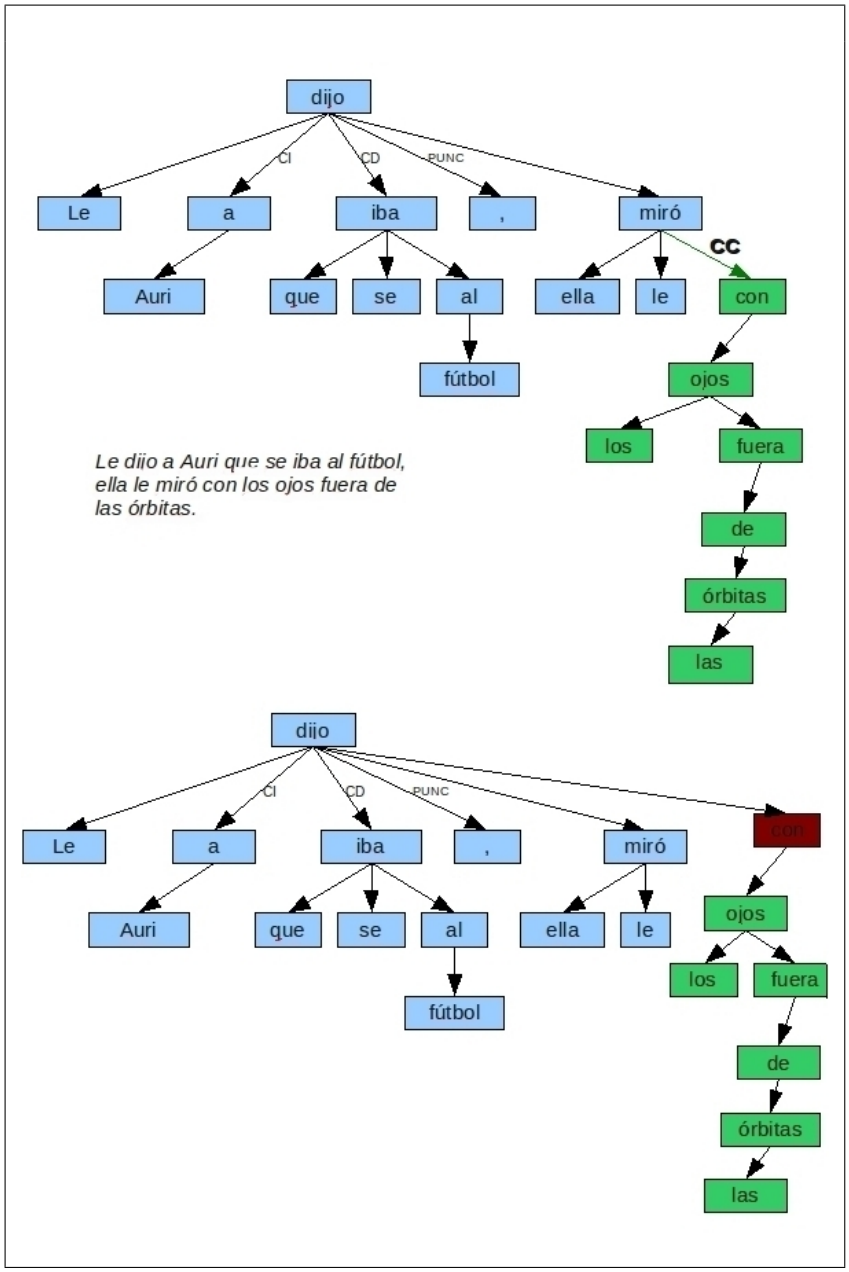

Figura 3.19: Ejemplo de un árbol de dependencias bien construido (arriba) mediante el analizador n–versiones y mal construido (abajo) por el analizador general. El análisis incorrecto de la preposición *con* provoca un error en cadena en toda una rama del árbol.

un análisis de dependencias de más calidad y con árboles resultantes mejor construidos, no tanto en cifras sino en estructura.

A partir de aquí pueden sugerirse numerosas ramas de investigación

para conseguir mejores resultados. Lo primero, que además es sugerencia obligada, consiste en buscar nuevos modelos de características capaces de establecer mejores parámetros para que los analizadores específicos que no obtuvieron un análisis ideal lo consigan. Lo segundo, que ya se ha comentado en esta sección, es buscar nuevas palabras y continuar con el experimento y así conseguir mejores resultados. Lo tercero, versa sobre la idea de construir de una manera diferente los corpora de entrenamiento y test, buscando algunas técnicas que permitan incluir el mismo porcentaje de frases de cada tipo (entendiendo por tipo los patrones que se describen en este experimento para cada palabra) en ambos corpora.

3.3. Algoritmo del Analizador N–Versiones

En esta Sección se describe un algoritmo implementado que automatiza la decisión de qué analizador específico concreto es mejor para analizar las palabras problemáticas. Sólo se ha realizado para la conjunción, pero puede ser extrapolable, obteniendo así un analizador N–versiones más general. El algoritmo se ha implementado en Perl[1] y realiza las siguientes tareas:

- 1. Analiza la frase con el analizador general.
- 2. Si detecta que existe una conjunción, mediante un sistema de reglas, decide qué analizador es más adecuado, según la cercanía a cierto tipo de palabras, o consultando información extra que el corpus AnCora nos ofrece.
- 3. Se analiza la frase con el analizador más adecuado decidiendo por la cercanía a la conjunción y el tipo de etiquetado que lleva consigo.
- 4. Se elimina el nodo del árbol resultante del analizador general.
- 5. Se introduce el nuevo nodo obtenido por el analizador específico.
- 6. Se devuelve la frase.

El algoritmo implementado no es capaz de decidir correctamente en todos los casos, ya que hay ciertas veces que la cercanía entre la conjunción y la palabra a la que debe ir conectada no es la mejor decisión. De hecho los resultados obtenidos son muy similares a los del analizador general. El analizador N–Versiones produce 57 fallos para la conjunción mientras que el analizador general produce 56 fallos. Es necesario un análisis detallado de ciertos aspectos de AnCora para alcanzar mayor fiabilidad, ya que en algunas palabras existe un alto grado de ambigüedad, porque existen análisis diferentes de frases con la misma estructura gramatical, por lo tanto, el

[1] www.perl.com

sistema no es capaz de tomar la decisión adecuada debido a las deficiencias del corpus. Con un corpus mejor etiquetado en ciertos casos ambiguos el mismo sistema devolvería unos resultados mejores.

3.4. Recapitulación

En este capítulo se han mostrado los experimentos realizados y resultados obtenidos con el objetivo de alcanzar una mejor precisión en el análisis sintáctico de dependencias. En primer lugar, un conjunto de experimentos de alto nivel, orientados al tamaño del corpus de entrenamiento, comprobando de esa manera, la homogeneidad de los cálculos. Acto seguido se muestran otros experimentos de bajo nivel, basados en el análisis de las palabras peor analizadas. Como consecuencia de estos experimentos de bajo nivel, se plantea un analizador N–Versiones consistente en varios analizadores específicos entrenados para analizar las palabras que provocan más fallos y un analizador general que se encarga del análisis del resto.

Capítulo 4

Conclusiones y Trabajo Futuro

En esta Capítulo se exponen las conclusiones generales derivadas de la realización de este proyecto, además se exponen las sugerencias de posible trabajo futuro.

4.1. Conclusiones

En esta Sección se exponen las conclusiones generales que se extraen de la realización de los experimentos realizados en este trabajo.

4.1.1. El Tamaño del Corpus de Entrenamiento es Relevante, pero no es la Única Vía

Como se expone en la Sección 3.1 y en (Ballesteros et al., 2010b), la creencia siempre ha sido que a más cantidad de ejemplos de entrenamiento, mejor calidad en el análisis. Como se puede ver en los resultados obtenidos por el analizador general, y tras ser mejorados por la aproximación consistente en un análisis n-versiones (Ballesteros et al., 2010a) y (Ballesteros et al., 2010c), usando diversos analizadores específicos entrenados mediante corpora muy pequeños. Se observa que no es necesario tener corpora muy grandes sino más bien corpora bien seleccionados para no incurrir en problemas de ambigüedad con el analizador y poder de esa manera obtener un mejor análisis.

Por tanto, y como se sugiere en la Sección 4.2, es interesante estudiar nuevas técnicas para la generación, o más bien diseño y construcción, de nuevos corpora de entrenamiento y análisis pensados para contener ejemplos que no provoquen ambigüedad en el análisis y decisiones incorrectas por parte del analizador.

4.1.2. El Analizador N-Versiones es una Propuesta Viable

Dados los resultados obtenidos en el Capítulo 3, tanto en los experimentos relacionados con la homegeneidad del análisis como en los resultados referentes al analizador n–Versiones se puede concluir que aún queda mucho esfuerzo por hacer para obtener una precisión mayor para el análisis de dependencias, ya que los buenos resultados globales pueden resultar engañosos si consideramos cuantas frases están perfectamente analizadas del corpus completo.

Por tanto, es interesante continuar la investigación en la vía de los analizadores n–versiones, ya que así se obtienen árboles de dependencias mucho mejor estructurados, cuando se focaliza el trabajo en ciertas palabras importantes dentro de cualquier frase. Con los resultados obtenidos se puede afirmar que esta aproximación es viable ya que hemos reducido el número de errores para las palabras problemáticas de manera bastante eficaz, con lo que se puede afirmar que es una técnica viable. Además es una técnica novedosa que nosotros hemos realizado de manera específica para el castellano pero que puede ser exportable para realizarla con un carácter más multilingüe.

4.2. Trabajo Futuro

Una vez concluido el trabajo surgen numerosas ideas orientadas a la búsqueda de obtener una mejor precisión o quizá una precisión de más calidad, ya que hay algunas palabra que provocan que el árbol de dependencias esté mal estructurado cuando se analizan incorrectamente, como se puede ver en la Sección 3.2.12. Tras nuestros esfuerzos, aun queda un 17,30 % de mejora necesaria para obtener un árbol de dependencias perfectamente etiquetado con cada frase que se analice. Existen numerosas áreas en las cuales podría realizarse trabajo para mejorar la precisión en el análisis.

1. Búsqueda de más palabras que tienen un análisis frecuentemente mal etiquetado: Esto implicaría continuar con el experimento de la Sección 3.2, una vez demostrada la viabilidad del mismo, mejorando de esta manera la precisión global y, como se concluye en el experimento, la precisión local de los árboles de dependencias, lo que es muy importante para muchos propósitos.

2. Incorporación de la idea del Analizador N–Versiones a un analizador concreto: Esta tarea, ya ha sido iniciada y sería de gran interés tener un producto final que consistiera en un analizador sintáctico de dependencias que incorpore las mejoras para el castellano propuestas, estudiadas y demostradas en este trabajo.

3. Nuevas técnicas para la construcción de corpora de entrenamiento y análisis: Como se puede observar en (Ballesteros et al., 2010b) y en la Sección 3.1 la precisión obtenida depende del tamaño del corpus de entrenamiento, pero también depende en buena medida de como este diseñado este corpus. Como se sugiere en (Ballesteros et al., 2010a) y (Ballesteros et al., 2010c), deben estudiarse nuevas formas de diseñar corpora para obtener un análisis mejor.

4. Buscar nuevas técnicas, modelos de entrenamiento capaces de mejorar los resultados obtenidos por nuestros analizadores específicos de manera que así se obtengan resultados mejores, en los casos que sea posible.

5. Extensión del trabajo a otras lenguas: Es factible, y practicamente obvio, que la aplicación de estas técnicas se puede exportar a cualquier lenguaje, susceptible de ser analizado mediante un analizador multilingüe como Maltparser. Puede ser de gran interés extender estas técnicas a otros idiomas, comprobar y corroborar que son también viables y no exclusivas del castellano. Dada la naturaleza de los experimentos, ya se observa que son de carácter general a pesar de haberse realizado de manera específica para un idioma concreto.

4.3. Conclusiones personales del Proyecto de Fin de Máster

Puedo afirmar que este Proyecto de Fin de Máster en Investigación en Informática hace justicia a su nombre en mi caso, ya que mi trabajo durante el periodo de investigación ha sido, bajo mi punto de vista, de auténtica investigación y estudio. He aprendido, dejando de lado los nuevos conocimien-

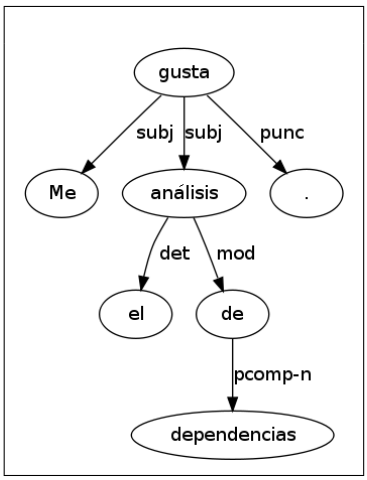

Figura 4.1: Me gusta el análisis de dependencias.

tos técnicos que el trabajo me ha aportado, a enfrentarme a tareas nuevas y totalmente desconocidas para mí, a enfrentarme a la presentación de un artículo en un congreso en un país extranjero y en un idioma diferente al mío, a escribir una memoria de un periodo de investigación que nunca pensé que fuera tan complicado y a tratar con tareas que pocos han abordado.

Aunque la Figura 4.1 pueda parecer banal, o algo irrisorio, considero que esta imagen resume perfectamente lo que ha sido la realización de este proyecto.

Y sobre todo, lo que para mí es más importante, me ha abierto una puerta, a un *mundo*, a un escenario, que me gustaría disfrutar y aprovechar, y espero y deseo que así sea en un futuro.

Apéndice A

Apéndices

A.1. Publicaciones derivadas del trabajo de Fin de Máster

Esta Tesis de Fin de Máster ha dado lugar a las siguientes publicaciones científicas.

- Ballesteros M, Herrera J, Francisco V, Gervás P. 2010. Improving Parsing Accuracy for Spanish using Maltparser. Revista sobre Procesamiento del Lenguaje Natural. 44:83-90 Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN) [1].
- Ballesteros M, Herrera J, Francisco V, Gervás P. 2010. A Feasibility Study on Low Level Techniques for Improving Parsing Accuracy for Spanish Using Maltparser. 6th Hellenic Conference on Artificial Intelligence. Athens. Greece. LNAI 6040:39-48. Springer Verlag.
- Ballesteros M, Herrera J, Francisco V, Gervás P. 2010. Towards a N-Version Dependency Parser. 13th International Conference on Text, Speech and Dialogue. Brno. Czech Republic. LNAI. Springer Verlag.

Además, el artículo *A Feasibility Study on Low Level Techniques for Improving Parsing Accuracy for Spanish Using Maltparser* fue admitido y presentado oralmente ante la comunidad científica en el 6th Hellenic Conference on Artificial Intelligence [2], que tuvo lugar entre los días 4 y 7 de mayo en la ciudad de Atenas, Grecia. La presentación tuvo lugar en el auditorio de la Fundación Eugenides[3] el 7 de mayo de 2010.

Igualmente, el artículo *Towards a N–Version Dependency Parser* ha sido aceptado y será presentado oralmente en la semana del 6 al 10 de septiembre

[1] http://www.sepln.org
[2] http://www.setn2010.gr
[3] http://www.eugenfound.edu.gr

de 2010 en el 13th International Conference on Text, Speech and Dialogue, en Brno, República Checa[4].

A.2. Resultados globales de la CoNLL–X Shared Task

En este apéndice se explican los resultados de la Tabla A.1. En la misma se observan los resultados obtenidos por cada grupo de investigación para cada lenguaje. Como se puede observar los resultados del castellano se encuentran en la media y son bastante buenos. Se muestran en negrita los resultados del grupor de Nivre et al. y los resultados del castellano.

- (1) Canisius et al. (Canisius et al., 2006).
- (2) Giuseppe Attardi (Attardi, 2006).
- (3) YuChieh Wu (Wu, Lee, y Yang, 2006).
- (4) Carreras et al. (Carreras, Surdeanu, y Màrquez, 2006).
- (5) Deniz Yuret (Yuret, 2006).
- (6) Eckhard Bick (Bick, 2006).
- (7) Nivre et al. (Nivre et al., 2006).
- (8) Michael Schiehlen (Schiehlen y Spranger, 2006).
- (9) Jinshan Ma et al. (Liu et al., 2006).
- (10) Dreyer et al. (Dreyer, Smith, y Smith, 2006).
- (11) Chang et al. (Chang, Do, y Roth, 2006).
- (12) Johansson et al.(Johansson y Nugues, 2006).
- (13) McDonald et al. (McDonald, Lerman, y Pereira, 2006).
- (14) Riedel et al. (Riedel, Çakici, y Meza-Ruiz, 2006).
- (15) Kenji Sagae et al. (Sagae et al., 2007).
- (16) Nobuyuki Shimizu (Shimizu, 2006).
- (17) Simon Corston-Oliver (Corston-Oliver y Aue, 2006).
- (18) Yuchang Cheng (Cheng, Asahara, y Matsumoto, 2006).

[4]http://www.tsdconference.org/tsd2010/index.html

A.3. Wiki sobre análisis de dependencias

Existe una wiki sobre análisis de dependencias en que contiene la experiencia producida en las dos CoNLL Shared Tasks basadas en análisis de dependencias, 2006 y 2007: http://depparse.uvt.nl/depparse-wiki/

Tabla A.1: Resultados generales obtenidos por los distintos grupos de investigación en la CoNLL–X Shared Task

Arabic	Chinese	Czech	Danish	Dutch	German	Japanese	Portuguese	Slovene	**Spanish**	Swedish	Turkish	AV	SD	Bulgarian	Name
57,64	78,37	60,92	77,90	74,59	77,56	87,41	77,42	59,19	**68,32**	79,15	51,07	70,80	11,11	78,74	(1) Canisius et al
53,81	54,89	59,76	66,35	58,24	69,77	65,38	75,36	57,19	**67,44**	68,77	37,80	61,23	9,92	72,89	(2) Giuseppe Attardi
63,81	74,81	59,36	78,38	68,45	76,52	90,11	81,47	67,83	**72,99**	71,72	55,09	71,71	9,67	79,73	(3) YuChieh Wu
60,94	83,68	68,82	79,74	67,25	82,41	88,13	83,37	68,43	**77,16**	78,65	58,06	74,72	9,72	83,30	(4) Carreras et al
52,42	72,72	51,86	71,56	62,75	63,82	84,35	70,35	55,06	**69,63**	65,23	60,31	65,01	9,46	73,49	(5) Deniz Yuret
55,37	76,18	63,02	74,61	69,51	74,74	84,75	78,18	64,31	**71,37**	74,09	53,87	70,00	9,25	79,21	(6) Eckhard Bick
66,71	**86,92**	**78,42**	**84,77**	**78,59**	**85,82**	**91,65**	**87,60**	**70,30**	**81,29**	**84,58**	**65,68**	**80,19**	**8,53**	**87,41**	**(7) Nivre et al**
44,39	66,20	53,34	76,05	72,11	68,73	83,35	71,01	50,72	**46,96**	71,10	49,81	62,81	13,01	0,00	(8) Michael Schiehlen
50,74	75,29	58,52	77,70	59,36	68,11	70,84	71,13	57,21	**65,08**	63,83	41,72	63,29	10,42	67,64	(9) Jinshan Ma et al
53,37	71,63	60,54	66,61	61,56	70,97	82,87	75,28	58,73	**67,62**	67,58	46,05	65,23	9,93	74,81	(10) Dreyer et al
60,92	85,05	72,88	80,60	72,91	84,17	89,07	83,99	69,52	**79,72**	82,31	60,51	76,80	9,43	0,00	(11) Chang et al
64,29	72,49	71,46	81,54	72,67	80,43	85,63	84,57	66,43	**78,16**	78,13	63,39	74,93	7,65	0,00	(12) Johansson et al
66,91	85,90	80,18	84,79	79,19	87,34	90,71	86,82	73,44	**82,25**	82,55	63,19	80,27	8,43	87,57	(13) McDonald et al
66,65	89,96	67,44	83,63	78,59	86,24	90,51	84,43	71,20	**77,38**	80,66	58,61	77,94	10,05	0,00	(14) Riedel et al
62,71	84,73	75,24	81,56	76,61	84,92	90,37	86,01	69,06	**77,68**	82,00	63,21	77,84	8,95	0,00	(15) Kenji Sagae et al
62,83	0,00	0,00	75,81	0,00	0,00	0,00	0,00	64,57	73,17	**79,49**	54,23	34,18	36,26	0,00	(17) Nobuyuki Shimizu
63,53	79,92	74,48	81,74	71,43	83,47	89,95	84,59	72,42	**80,36**	79,69	61,74	76,94	8,47	83,36	(17) Simon Corston
65,19	84,27	76,24	81,72	71,77	84,11	89,91	85,07	71,42	**80,46**	81,08	61,22	77,70	8,67	86,34	(18) Yuchang Cheng
59,94	78,32	67,17	78,31	70,73	78,58	85,86	80,63	65,16	**73,52**	76,44	55,95			79,98	Average

Lista de tablas

Índice de figuras

References

Afonso, Susana, Eckhard Bick, Renato Haber, y Diana Santos. 2002. Floresta sintá(c)tica: A treebank for Portuguese. En *Third International Conference on Language Resources and Evaluation (LREC 2002) (Las Palmas de Gran Canaria, Spain)*.

Attardi, Giuseppe. 2006. Experiments with a multilanguage non-projective dependency parser. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 166–170, Morristown, NJ, USA. Association for Computational Linguistics.

Ballesteros, Miguel, Jesús Herrera, Virginia Francisco, y Pablo Gervás. 2010a. A feasibility study on low level techniques for improving parsing accuracy for spanish using maltparser. En Stasinos Konstantopoulos Stavros Perantonis Vangelis Karkaletsis Constantine D. Spyropoulos, y George Vouros, editores, *Artificial Intelligence: Theories, Models and Applications*, volumen 6040 de *Lecture Notes in Artificial Intelligence*, páginas 39–48. Springer.

Ballesteros, Miguel, Jesús Herrera, Virginia Francisco, y Pablo Gervás. 2010b. Improving Parsing Accuracy for Spanish using Maltparser. *Journal of the Spanish Society for NLP (SEPLN)*, 44:83–90, 05/2010.

Ballesteros, Miguel, Jesús Herrera, Virginia Francisco, y Pablo Gervás. 2010c. Towards a N-Version Dependency Parser. *In press. 13th International Conference on Text, Speech and Dialogue.*

Ballesteros, Miguel, Raúl Martín, y Belén Díaz Agudo. 2010. JadaWeb: A CBR System for Cooking Recipes. En *Proceedings of Workshop on Computer Cooking Contest (ICCBR 2010)*, Alessandria Italy.

Bick, Eckhard. 2006. Lingpars, a linguistically inspired, language-independent machine learner for dependency treebanks. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 171–175, Morristown, NJ, USA. Association for Computational Linguistics.

Böhmová, Alena, Jan Hajič, Eva Hajičová, y Barbora Hladká. 2001. The Prague Dependency Treebank: Three-level annotation scenario. En Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Buch-Kromann, Mathias. 2005. Discontinuous grammar. a model of human parsing and language acquisition. Phd Thesis. Copenhaguen Business School, Copenhagen, Denmark. 2005.

Buchholz, Sabine y Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 149–164, Morristown, NJ, USA. Association for Computational Linguistics.

Canisius, Sander, Toine Bogers, Antal Van Den Bosch, y Jeroen Geertzen. 2006. Dependency parsing by inference over high-recall dependency predictions. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*.

Carreras, Xavier, Mihai Surdeanu, y Lluís Màrquez. 2006. Projective dependency parsing with perceptron. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 181–185, Morristown, NJ, USA. Association for Computational Linguistics.

Caseli, Helena M., Tiago F. Pereira, Lucia Specia, Thiago A. S. Pardo, Caroline Gasperin, y Sandra M.Aluisio. 2009. Building a Brazilian Portuguese Parallel Corpus of Originial and Simplified Texts. En *Proceedings of CiCLing*.

Chang, Ming Wei, Quang Do, y Dan Roth. 2006. A pipeline model for bottom-up dependency parsing. En *Journal of the School of Engineering, The University of Tokyo*, páginas 15–40.

Cheng, Yuchang, Masayuki Asahara, y Yuji Matsumoto. 2006. Multi-lingual dependency parsing at NAIST. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 191–195, Morristown, NJ, USA. Association for Computational Linguistics.

Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies. En *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, páginas 184–191, Morristown, NJ, USA. Association for Computational Linguistics.

Corston-Oliver, Simon y Anthony Aue. 2006. Dependency parsing with reference to Slovene, Spanish and Swedish. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 196–200, Morristown, NJ, USA. Association for Computational Linguistics.

Crammer, Koby y Yoram Singer. 2001. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:2003.

Dreyer, Markus, David A. Smith, y Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. En *CoNLL-X '06:*

Proceedings of the Tenth Conference on Computational Natural Language Learning, páginas 201–205.

Eisner, Jason. 1996. Three new probabilistic models for dependency parsing: An exploration. En *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, páginas 340–345, Copenhagen.

Foth, Kilian A., Michael Daum, y Wolfgang Menzel. 2004. Interactive grammar development with WCDG.

Gambäck, Björn, Fredrik Olsson, Atelach Alemu, y Argaw Lars Asker. 2009. Methods form amharic part-of-speech tagging.

Gelbukh, Alexander, Sulema Torres, y Hiram Calvo. 2005. Transforming a constituency treebank into a dependency treebank. En *Procesamiento Lenguaje Natural*.

Hajič, Jan y Petr Zemánek. 2004. Prague arabic dependency treebank: Development in data and tools. En *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, páginas 110–117.

Herrera, J. y P. Gervás. 2008. Towards a Dependency Parser for Greek Using a Small Training Data Set. *Journal of the Spanish Society for Natural Language Processing (SEPLN)*, 41:29–36.

Herrera, Jesús, Pablo Gervás, P.J. Moriano, Alfonso Moreno, y Luis Romero. 2007a. Building corpora for the development of a dependency parser for spanish using maltparser. *Revista de la Asociación Española para el Procesamiento del Lenguaje Natural*, 39:181–186, 09/2007.

Herrera, Jesús, Pablo Gervás, P.J. Moriano, Alfonso Moreno, y Luis Romero. 2007b. Jbeaver: Un analizador de dependencias para el español. *Revista de la Asociación Española para el Procesamiento del Lenguaje Natural*, 39:285–286, 09/2007.

Herrera, Jesús, Anselmo Peñas, y Felisa Verdejo. 2005. Textual Entailment Recognition based on Dependency Analysis and Wordnet. En *Proceedings of the 1st. PASCAL Recognision. Textual Entailment Challenge*.

Johansson, Richard y Pierre Nugues. 2006. Investigating multilingual dependency parsing. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 206–210, Morristown, NJ, USA. Association for Computational Linguistics.

Kromann, Mathias. 2003. The Danish dependency treebank and the dtag treebank tool. En *2nd Workshop on Treebanks and Linguistic Theories (TLT), Sweden*, páginas 217–220.

Kübler, Sandra, Ryan T. McDonald, y Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Kübler, Sandra, Jelena Prokić, y Rijksuniversiteit Groningen. 2006. Why is German dependency parsing more reliable than constituent parsing. En *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT*, páginas 7–18.

Kudo, Taku y Yuji Matsumoto. 2002. Japanese dependency analyisis using cascaded chunking. En *Proceedings of the 6th Workshop on Computational Languager Learning (CoNLL)*.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. En *Proc. Workshop on the Evaluation of Parsing Systems*, Granada.

Liu, Ting, Jinshan Ma, Huijia Zhu, y Sheng Li. 2006. Dependency parsing based on dynamic local optimization. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 211–215, Morristown, NJ, USA. Association for Computational Linguistics.

Mariona Taulé, M. Antònia Martí y Marta Recasens. 2008. Ancora: Multilevel annotated corpora for Catalan and Spanish. En *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA).

McDonald, R., K. Lerman, y F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. En *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL–X)*, páginas 216–220.

McDonald, Ryan y Joakim Nivre. 2007. Characterizing the errors of data–driven dependency parsing models. En *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, páginas 122–131. Association for Computational Linguistics.

Morante, Roser, Anthony Liekens, y Walter Daelemans. 2008. Learning the scope of negation in biomedical texts. En *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, páginas 715–724, Morristown, NJ, USA. Association for Computational Linguistics.

Navarro, Borja, Montserrat Civit, M^{a} Antonia Martí, Raquel Marcos, y Belén Fernández. 2003. Syntactic, semantic and pragmatic annotation

in Cast3LB. En *Proceedings of Workshop on Shallow Processing of Large Corpora, March 27 (SProLaC03)*, Lancaster, UK.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan Mcdonald, Jens Nilsson, Sebastian Riedel, y Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. En *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, páginas 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

Nivre, Joakim, Johan Hall, y Jens Nilsson. 2004. Memory–based dependency parsing. En *Proceedings of CoNLL–2004*, páginas 49–56. Boston, MA, USA.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kbler, Svetoslav Marinov, y Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Nivre, Joakim, Johan Hall, Jens Nilsson, G. Eryiğit, y S. Marinov. 2006. Labeled pseudo–projective dependency parsing with support vector machines. En *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL–X)*, páginas 221–225.

Riedel, Sebastian, Ruket Çakici, y Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 226–230, Morristown, NJ, USA. Association for Computational Linguistics.

Sagae, Kenji, Eric Davis, Alon Lavie, Brian MacWhinney, y Shuly Wintner. 2007. High-accuracy annotation and parsing of CHILDES transcripts. En *CACLA '07: Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, páginas 25–32, Morristown, NJ, USA. Association for Computational Linguistics.

Sampson, Geoffrey. 2002. Briefly noted - english for the computer: the susanne corpus and analytic scheme. *Comput. Linguist.*, 28(1):102–103.

Schiehlen, Michael y Kristina Spranger. 2006. Language independent probabilistic context-free parsing bolstered by machine learning. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 231–235, Morristown, NJ, USA. Association for Computational Linguistics.

Shimizu, Nobuyuki. 2006. Maximum spanning tree algorithm for non-projective labeled dependency parsing. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*,

páginas 236–240, Morristown, NJ, USA. Association for Computational Linguistics.

Sjöbergh, Jonas. 2003. Combining pos-taggers for improved accuracy on Sweedish text. En *Proceedings of Nodaliba 2003*, Reykjavik, Iceland.

Wu, Yu-Chieh, Yue-Shi Lee, y Jie-Chi Yang. 2006. The exploration of deterministic and efficient dependency parsing. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 241–245, Morristown, NJ, USA. Association for Computational Linguistics.

Yamada, H. y Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. En *Proceedings of International Workshop of Parsing Technologies (IWPT'03)*, páginas 195–206.

Yuret, Deniz. 2006. Dependency parsing as a classification problem. En *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning*, páginas 246–250, Morristown, NJ, USA. Association for Computational Linguistics.

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster:"Mejora de la precisión para el análisis de dependencias usando Maltparser para el Castellano", realizado durante el curso académico 2009-2010 bajo la dirección de Pablo Gervás [y con la colaboración externa de dirección de Virginia Francisco y Jesús Herrera] en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Miguel Ballesteros Martínez
Fdo: