JBeaver: Un Analizador de Dependencias para el Español Basado en Aprendizaje

Jesús Herrera†, Pablo Gervás‡, Pedro J. Moriano‡, Alfonso Muñoz‡, Luis Romero‡

†Departamento de Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia
C/ Juan del Rosal, 16, E-28040 Madrid
jesus.herrera@lsi.uned.es
‡Departamento Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
C/ Profesor José García Santesmases, s/n, E-28040 Madrid
pgervas@sip.ucm.es, {pedrojmoriano, alfonsomm, luis.romero.tejera}@gmail.com

Resumen En el presente artículo se describe un proyecto de investigación aplicada en el que, haciendo uso de una herramienta de aprendizaje automático específica (*Maltparser*), se ha desarrollado un analizador de dependencias para el español. Este analizador se caracteriza por ser públicamente disponible y aunar, así mismo, las siguientes características: autónomo, fácil de instalar y de utilizar (mediante interfaz gráfica o por comandos de consola) y de elevada precisión. Además, el sistema desarrollado sirve para entrenar de manera sencilla modelos de *Maltparser*, por lo que se configura en potencia como un analizador de dependencias para cualquier idioma.

1. Introducción

El análisis sintáctico de dependencias es una técnica que ha experimentado un notable auge, en los últimos tiempos, en diferentes campos del Procesamiento del Lenguaje Natural. Esto se ha podido observar en ciertas tareas internacionales, como el *Recognizing Textual Entailment Challenge*[1] de la Red de Excelencia PASCAL[2] o la tarea de Pregunta–Respuesta del *Cross Language Evaluation Forum*[3], en las que el uso del análisis de dependencias ha mostrado ser una herramienta eficaz. Así mismo, la disponibilidad pública y gratuita de algunos analizadores de dependencias de alta eficiencia, como es el caso de *Minipar* [10], han propiciado que muchos equipos de investigación decidiesen incorporar a sus sistemas un módulo de análisis de dependencias. De este interés en el uso

[1] http://www.pascal-network.org/Challenges/RTE/
[2] http://www.pascal-network.org/
[3] http://clef-qa.itc.it/

de analizadores de dependencias se derivan eventos internacionales específicos, como la *CoNLL-X Shared Task: Multilingual Dependency Parsing*[4] [1].

Entre los analizadores de dependencias existentes es de reseñar el caso de *Minipar*; este analizador se caracteriza por su alto rendimiento en el análisis y alta eficiencia temporal, así como por su disponibilidad pública y gratuita. Todas estas características han sido un estímulo para que los grupos de investigación lo utilizasen cada vez con mayor profusión. El éxito de *Minipar* en las tareas a las que se podía aplicar, es decir, las proyectadas para lengua inglesa, hizo desear a los equipos de investigación implicados la disponibilidad de analizadores de dependencias para lenguas distintas al inglés. El paso al multilingüismo en las aplicaciones en las que estaban inmersos se vería facilitado si pudiesen contar con analizadores similares a *Minipar* que funcionasen para otras lenguas.

El problema surgía ahora al plantearse el desarrollo de un analizador de dependencias para un idioma: ¿sería posible conseguirlo a bajo coste y en poco tiempo? La respuesta venía de manos de los generadores de analizadores de dependencias, por ejemplo *Maltparser* [12]. Con *Maltparser*, la idea de obtener un analizador de dependencias para una lengua determinada se reduce a la aparentemente simple cuestión de entrenarlo con un corpus etiquetado con análisis de dependencias. El paso de este mundo de las ideas al mundo de los hechos se describe en el presente artículo para un caso concreto: el desarrollo de un analizador de dependencias para el español.

2. Revisión del Trabajo Previo y Motivaciones para Construir *JBeaver*

Entre los trabajos clásicos en sistemas de análisis de dependencias es indiscutible señalar a *Minipar* como un referente fundamental para el inglés. *Minipar* ha sido una herramienta profusamente utilizada en el ámbito del Procesamiento del Lenguaje Natural, como lo demuestran las 170 citas del artículo de Dekang Lin [10] que registra actualmente el sistema *Google Scholar*[5].

El amplio interés que el análisis de dependencias ha suscitado en la comunidad científica, claramente influido por las notables características de *Minipar*, ha llevado a querer construir analizadores de dependencias para cada vez más lenguas. Aprovechando las facilidades que para esta labor proporcionan los sistemas de aprendizaje automático, recientemente se han desarrollado herramientas como *Maltparser*. Este tipo de sistemas son, realmente, generadores de analizadores de dependencias. Con los generadores de analizadores que basan su funcionamiento en el aprendizaje, se pueden obtener analizadores para cualquier lengua para la que se disponga de un corpus etiquetado con análisis de dependencias. En concreto, *Maltparser* es un sistema de análisis guiado por los datos con pretensión de ser independiente del idioma. Su creación está motivada por el hecho de que los analizadores de dependencias que están demasiado ajustados

[4] `http://nextens.uvt.nl/~conll/`
[5] `http://scholar.google.com`

a un cierto idioma funcionan mal para otros; además, se busca también en él una solución para el indeterminismo de los analizadores de dependencias [12]. Para la fase de aprendizaje *Maltparser* incorpora dos posibilidades a elegir por el usuario, ambas dentro del ámbito del aprendizaje supervisado: bien mediante máquina de vector de soporte o bien mediante un modelo basado en memoria.

La existencia de *Minipar* y la de *Maltparser* suponían un estímulo importante para querer desarrollar un analizador de dependencias para el español. Gracias a *Maltparser* era viable, sin muchos recursos, la realización de un programa de características similares a las de *Minipar*, que tan buenos resultados estaba dando, que realizase análisis de dependencias para el español.

Pero, además, había otro hecho que motivaba tal desarrollo: en el momento de iniciar el proyecto *JBeaver* ya existían analizadores de dependencias para lenguas no muy extendidas, como el sueco [12] o el turco [7]. Sin embargo, por entonces no existía un analizador de dependencias para el español, la tercera lengua más utilizada en el mundo. Por el contrario para las dos lenguas de uso más frecuente, el chino [9] y el inglés [10], ya se habían desarrollado analizadores de este tipo. Además, es notable el interés por equiparar los recursos disponibles para todas las lenguas europeas, como se puede observar en campañas como el Cross Language Evaluation Forum. Era, pues, el momento de abordar esta tarea.

Otra muestra del interés suscitado en este campo se encuentra en foros internacionales en los que se evalúan analizadores de dependencias para múltiples lenguas, como puede ser la *CoNLL-X Shared Task: Multilingual Dependency Parsing*. En este tipo de eventos se pueden probar las posibilidades que tienen los generadores de analizadores, como *Maltparser*, para producir analizadores válidos para diversas lenguas; pero no sólo eso, sino que se pueden también evaluar las posibilidades de otros enfoques alternativos para abordar el problema. De este modo se ha estimulado la obtención de analizadores de dependencias para una amplia gama de lenguas; por ejemlo, en la tarea citada de la *CoNLL-X* se han estudiado las posibilidades actuales de realizar análisis de dependencias para los siguientes idiomas: danés, holandés, portugués, sueco, árabe, checo, búlgaro, español, alemán, japonés, esloveno, chino y turco. *Maltparser* otuvo en este foro unos resultados notables, y en concreto en el análisis del español [13], para el se obtuvieron valores muy satisfactorios, del orden de los mejores obtenidos por el propio sistema y, en cualquier caso, superiores a la media de los presentados a la tarea[6].

Posteriormente a la decisión de desarrollar *JBeaver*, algunos grupos de investigación dieron a conocer trabajos sobre analizadores de dependencias para el español, como los de Calvo y Gelbuckh [2], Canisius *et al.* [3], Carreras *et al.* [4] Corson y Aue [5] o Nivre *et al.* [13]; la mayor parte de ellos en el ámbito de la *CoNLL-X Shared Task: Multilingual Dependency Parsing*. Sin embargo, la existencia de estos sistemas no impidió continuar con el proyecto *JBeaver*, a pesar de que su enfoque era muy similar al propuesto por Nivre *et al.* [13]; de hecho, las buenas características demostradas por el modelo de Nivre *et al.* probaban que el camino elegido para el desarrollo de *JBeaver* era prometedor. Esta

[6] `http://w3.msi.vxu.se/users/jha/conllx/`

última consideración, unida a que todos los sistemas participantes en la *CoNLL-X Shared Task: Multilingual Dependency Parsing* eran prototipos de laboratorio no disponibles públicamente, permitieron centrar esfuerzos en conseguir una herramienta autónoma, disponible pública y gratuitamente, de fácil instalación y uso, y que permitiese la incorporación de un analizador de dependencias para el español a todos los proyectos interesados en estas técnicas.

Un factor importante para el desarrollo de analizadores de dependencias es la existencia de corpora anotados con los que apoyar esta labor. En concreto, para el español es reseñable la existencia del corpus *Cast3LB* [11], que ha sido utilizado tanto en el presente proyecto [8] como en la *CoNLL-X Shared Task: Multilingual Dependency Parsing*.

3. *JBeaver*

El objetivo final era un analizador de dependencias para el español, de libre distribución y que fuera fácil de instalar y manejar. Por otra parte, se debían acotar los esfuerzos dada la limitación de recursos del proyecto.

3.1. Decisiones de Diseño y Elección de Recursos

Bajo los requisitos del proyecto era inviable el desarrollo de la algorítmica propia del análisis de dependencias, por lo que se hubieron de buscar recursos que evitasen esta labor. Uno de ellos es *Maltparser* [12], que finalmente fue el elegido por las características que ofrecía: era autónomo, fácil de integrar como subsistema y proporcionaba unos resultados notables en las lenguas para las que se había probado hasta el momento.

Tanto para el entrenamiento de *Maltparser* como para la ejecución como analizador del modelo aprendido es necesario proporcionar el etiquetado de categorías gramaticales de las palabras del texto. Como uno de los objetivos era que *JBeaver* pudiese recibir textos sin anotar, para facilitar al máximo su uso, la propia herramienta debería etiquetar los textos recibidos a la entrada con su categoría gramatical. Igualmente que en el caso del análisis de dependencias, tampoco era factible el desarrollo de algoritmos para el etiquetado de categorías gramaticales. Por ello, fue necesario buscar una herramienta disponible, autónoma, fiable y fácil de integrar en *JBeaver*; esta fue, finalmente, *TreeTagger* [8] [14].

Así mismo, tanto el entrenamiento de *Maltparser* como la evaluación del producto final obtenido requieren corpora convenientemente anotados. Una vez más fue necesaria la búsqueda de recursos que evitasen tener que generar manualmente corpora con un volumen adecuado de análisis de dependencias. Este aspecto se vio resuelto con el uso del corpus *Cast3LB*, que contiene textos en español anotados con sus análisis sintácticos de constituyentes. Para obtener los corpora adecuados para el entrenamiento de *Maltparser* y la evaluación de *JBeaver*, se desarrolló una herramienta para convertir los análisis de constituyentes del *Cast3LB* en análisis de dependencias [8].

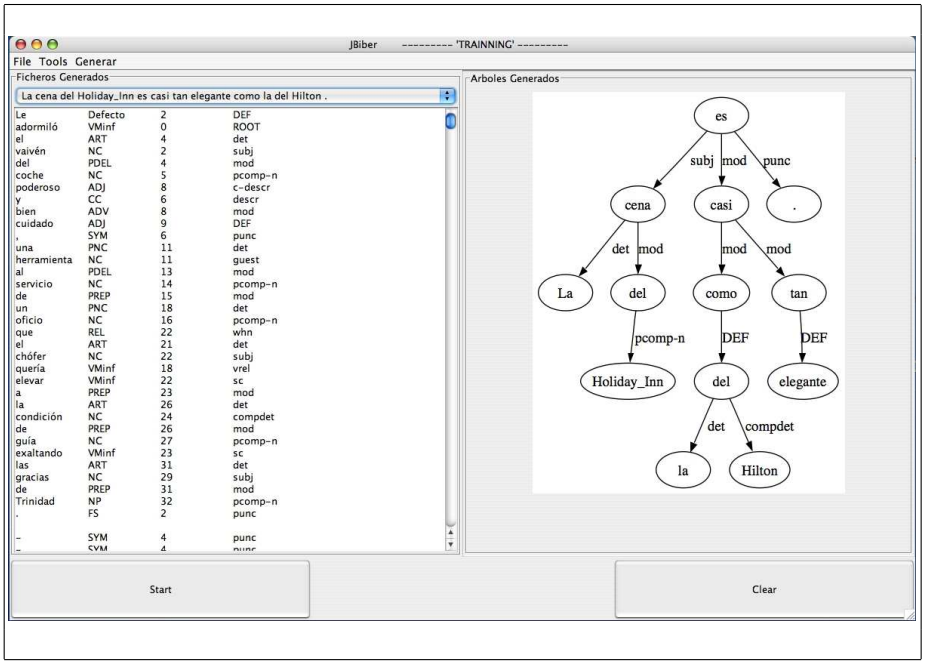

Figura 1. Interfaz gráfica de *JBeaver*

Otro aspecto definitorio de *JBeaver* es su interfaz gráfica de usuario (ver figura 1). En ésta se muestran los análisis obtenidos en forma de grafos, para que los datos resulten visualmente cómodos de interpretar. No obstante, también se proporciona la salida en forma de fichero de texto, para que pueda ser fácilmente manipulado por otros programas. La representación de los grafos quedó delegada a *Graphviz* [6], como otro de los subsistemas que forman parte de *JBeaver*.

Una vez evaluados los recursos disponibles y elegidos *Maltparser*, *TreeTagger*, *Cast3LB* y *Graphviz*, se podía dar comienzo al desarrollo de *JBeaver*, cuyo producto final sería distribuido para su uso con importantes sistemas operativos del mercado: *Linux* y *Apple Mac OS X*.

3.2. Desarrollo

El punto más importante a desarrollar en *JBeaver*, para tener un analizador de dependencias, era el entrenamiento de *Maltparser*. Pero, a su vez, había que construir una capa de software que envolviese e interrelacionase el modelo aprendido, el etiquetador morfológico y la entrada/salida con el usuario.

Entrenamiento del Núcleo Analizador

Con el entrenamiento de *Maltparser* se tendría un modelo que, incluido como módulo de *JBeaver*, actuaría como núcleo del análisis de dependencias. En virtud

de los resultados obtenidos por Nivre para el español [13], se seleccionó en la versión 4.0 de *Maltparser* la máquina de vector de soporte como algoritmo de aprendizaje.

Para el entrenamiento se proporcionó a *Maltparser* un corpus de análisis de dependencias [8] en el que cada palabra fuese etiquetada con: su categoría gramatical, un puntero a la palabra que esta modifica y la relación sintáctica entre ambas palabras. Para obtener este corpus se aplicó un algoritmo de conversión de árboles de análisis de constituyentes a árboles de análisis de dependencias al corpus *Cast3LB*. Una vez transformados los análisis, se les dio el formato adecuado para el entrenamiento de *Maltparser*. Para ello hubo que transformar las etiquetas extraídas del corpus *Cast3LB* en las adecuadas según los propósitos del proyecto. De este modo, las etiquetas de *Cast3LB* que indicaban relación sintáctica fueron sustituidas por las equivalentes propias de *Minipar*; así mismo, las etiquetas de categoría gramatical se sustituyeron por las equivalentes de *TreeTagger*. Estas transformaciones de etiquetas se justifican porque, en el caso de las categorías gramaticales, el modelo había de ser entrenado de manera congruente a como iba a ser utilizado posteriormente, es decir, los textos a analizar (introducidos por el usuario) serían etiquetados por *TreeTagger* actuando como subsistema de *JBeaver*; en el caso de las relaciones sintácticas el conjunto de etiquetas a utilizar por *JBeaver* no estaba restringido, por lo que se decidió usar las de *Minipar*, para facilitar el uso de *JBeaver* por parte de otros sistemas concebidos para usar *Minipar*.

El entrenamiento del modelo de *Maltparser* distribuido con *JBeaver* se realizó a partir de un conjunto de textos del corpus *Cast3LB* conformado por 22.413 palabras. El proceso de transformación de análisis de constituyentes en análisis de dependencias, el etiquetado del corpus de entrenamiento y el entrenamiento en sí se ejecutaron en pocos segundos en un computador personal.

El Software de Interrelación entre Subsistemas

En paralelo a la creación del corpus de análisis de dependencias y a la realización del entrenamiento, se desarrolló el software que gestionaba el funcionamiento ordenado de *Maltparser*, *TreeTagger* y la interfaz de usuario. En este punto se configuró *JBeaver* no sólo como un simple analizador de dependencias, sino que también se le dotó con la capacidad de servir como entrenador de modelos de *Maltparser* de uso sencillo mediante una interfaz gráfica. Además, funcionando en modo analizador puede ser invocado desde la consola de comandos del sistema operativo, inhibiéndose la ejecución de la interfaz gráfica de usuario y proporcionando, de esta manera, un funcionamiento similar al de *Minipar*.

Así pues, el software de interrelación entre los diferentes subsistemas de *JBeaver* permite:

1. Que el programa se ejecute sólo como analizador, usando el modelo de *Maltparser* que fue entrenado durante el desarrollo del proyecto y sin ejecutar la interfaz gráfica de usuario. En modo analizador, se invoca adecuadamente desde la consola de comandos del sistema operativo, proporcionándole el

fichero con el texto a analizar. La salida de *JBeaver*, en este caso, es exclusivamente en forma de fichero de texto.

2. Que el programa se pueda ejecutar bien como analizador con salida gráfica bien como entrenador de modelos de *Maltparser*. Para ello, desde la interfaz gráfica se pueden introducir textos para ser analizados, cuyo árbol de análisis es representado gráficamente por pantalla. En el otro modo de funcionamiento, se pueden introducir tanto corpora de entrenamiento como los parámetros de configuración de *Maltparser* para ser entrenado; de este modo se pueden obtener tantos modelos como se quiera para ser utilizados posteriormente como analizadores. Esta última posibilidad permitiría, virtualmente, que *JBeaver* pudiera ser utilizado para el análisis de dependencias en cualquier idioma, en función del modelo entrenado.

3.3. Pruebas

De las diversas pruebas a que fue sometido *JBeaver* durante la fase de desarrollo, son de destacar las relacionadas con el rendimiento del núcleo analizador, es decir, del modelo entrenado de MaltParser. Para ello se realizaron dos experimentos de medición basados en la propuesta de la *CoNLL-X Shared Task: Multilingual Dependency Parsing*, en los que se calcularon las métricas Labeled Attachment Score (LAS), Unlabeled Attachment Score (UAS) y Label Accuracy (LA) [1].

Para el primer experimento de medición, se seleccionó una fracción del corpus *Cast3LB*, de 20859 palabras, no usada previamente para el entrenamiento del modelo de *Maltparser* y se generó a partir de ella un corpus con análisis de dependencias, que se tomó como modelo de referencia. Se extrajeron los textos sin etiquetar de ese corpus y se sometieron al análisis de dependencias efectuado por el modelo aprendido. Posteriormente se comprobó la salida proporcionada por el analizador con el modelo de referencia, obteniéndose los siguientes valores: LAS = 74,32 %, UAS = 80,72 %, LA = 89,28 %.

El segundo experimento fue similar al primero, pero con textos no pertenecientes al *Cast3LB*, de los siguientes dominios: política, medicina, economía, justicia y deporte. Posteriormente, los textos fueron procesados con *JBeaver*, arrojando los siguientes resultados:

- Textos de política (548 palabras): LAS = 64,61 %, UAS = 73,56 %, LA = 82,31 %.
- Textos de medicina (313 palabras): LAS = 67,38 %, UAS = 76,6 %, LA = 83,33 %.
- Textos de economía (997 palabras): LAS = 66,63 %, UAS = 76,2 %, LA = 85,54 %.
- Textos de justicia (1726 palabras): LAS = 60,26 %, UAS = 73,17 %, LA = 76,83 %.
- Textos de deporte (596 palabras): LAS = 61,45 %, UAS = 76,73 %, LA = 78,36 %.

4. Conclusiones y Trabajo Futuro

JBeaver, el resultado del proyecto, es un sistema autónomo, disponible gratuitamente[7] para su ejecución en los principales sistemas operativos, fácil de instalar y utilizar, y con alto rendimiento para el análisis de dependencias del español. Al estar implementado en *Java*, ha de estar previamente instalado en el computador, el *Java Runtime Envorinment 5.0*[8]. Así mismo, es necesaria la instalación previa de *Graphviz*. Aunque el enfoque de la funcionalidad como analizador de dependencias de *JBeaver* es muy similar al del sistema desarrollado por el grupo de Nivre [13], ya que *JBeaver* hace uso de *Maltparser*, tres son las principales aportaciones que diferencian a *JBeaver*:

1. No es un prototipo de laboratorio, sino que está disponible públicamente en un paquete *jar*, muy simple de instalar y utilizar.
2. No sólo realiza análisis de dependencias de textos, mediante el modelo previamente entrenado de *Maltparser* incluido en la distribución, sino que cada usuario puede entrenar un modelo de *Maltparser* con el que ejecutar posteriormente análisis desde el propio *JBeaver*.
3. Posee una interfaz gráfica de usuario para el entrenamiento de *Maltparser* y para el análisis de textos, con salida visual de los árboles de análisis generados. Pero, además, puede ser invocado en modo no gráfico, desde la consola de comandos del sistema operativo, realizando la entrada/salida mediante ficheros de texto, y analizando mediante el modelo de *Maltparser* que incluye la distribución.

Así mismo, por los resultados obtenidos en las pruebas de evaluación a que ha sido sometido, se puede considerar a *JBeaver* como una interesante opción para el análisis de dependencias para el español.

Además, *JBeaver* es virtualmente un analizador de dependencias para cualquier idioma, ya que se puede utilizar para entrenar modelos de *Maltparser* que posteriormente pueden ser utilizados para el análisis. Pero hay que tener en cuenta que aunque *Maltparser* ha sido ya probado para varios idiomas, no se puede asegurar todavía su universalidad como analizador de dependencias. De este modo, una de las previsiones de desarrollo futuro para *JBeaver* es entrenarlo para el análisis de otras lenguas, adecuándolo a características propias que pudieran provocar incompatibilidades con el software actual, como pudieran ser los juegos de caracteres.

En cuanto al análisis del español, se ha de trabajar en buscar las causas y las posibles soluciones a los análisis defectuosos. Para ello habrá que someter a *JBeaver* a pruebas sistemáticas y específicas para determinados casos complicados como, por ejemplo, el análisis de la subordinación y la yuxtaposición. Dada la dependencia que *JBeaver* tiene de *Maltparser*, la corrección de estos errores se habrá de centrar en la elaboración de corpora de entrenamiento específicos, así como en el ajuste en la parametrización del sistema de aprendizaje.

[7] `http://nil.fdi.ucm.es/nilweb/`
[8] `http://java.sun.com/javase/downloads/index_jdk5.jsp`

Agradecimientos

Esta investigación ha sido parcialmente sufragada con cargo a los presupuestos del proyecto TIN2006-14433-C02-01, del Ministerio de Educación y Ciencia, así como por la subvención UCM-CAM-910494 otorgada al grupo de investigación de manera conjunta por la Universidad Complutense de Madrid y la Comunidad Autónoma de Madrid.

Referencias

1. A. Buchholz and E. Marsi. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA*, June 2006.
2. H. Calvo and A. Gelbukh. DILUCT: An Open–Source Spanish Dependency Parser based on Rules, Heuristics, and Selectional Preferences. In J. J. Katz, editor, *Proceedings of the 11th International Conference on Applications of Natural Language to Information Systems, NLDB 2006, Klagenfurt, Austria*, pages 164–175, May 2006.
3. A. Canisius, T. Bogers, A. van den Bosch, J.Geertzen, and E.T. Kin Sang. Dependency Parsing by Inference over High-recall Dependency Predictions. In *Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA*, June 2006.
4. X. Carreras, M. Surdeanu, and L. Màrquez. Projective Dependency Parsing with Perceptron. In *Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA*, April 2006.
5. S. Corston-Oliver and A. Aue. Dependency Parsing with Reference to Slovene, Spanish and Swedish. In *Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA*, June 2006.
6. J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North, and G. Woodhull. Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools. In M. Junger and P. Mutzel, editors, *Graph Drawing Software, Springer–Verlag, Berlin*, pages 127–148, 2003.
7. G. Eryiğit and K. Oflazer. Statistical Dependency Parsing of Turkish. In *Proceedings of EACL'06, Trento, Italy*, pages 89–96, 2006.
8. J. Herrera, P.Gervás, P.J. Moriano, A. Muñoz, and L. Romero. Building Corpora for the Development of a Dependency Parser for Spanish Using Maltparser. In *Actas de la Conferencia de la Sociedad Española para el Procesamiento del Lenguaje Natural, SEPLN 2007, Sevilla, España*, 2007.
9. M. Jinshan, Z. Yu, L. Ting, and L. Sheng. A Statistical Dependency Parser of Chinese under Small Training Data. In *Proceedings of the First International Joint Conference on Natural Language Processing, IJCNLP-04, Sanya City, Hainan Island, China*, March 2004.
10. D. Lin. Dependency–based Evaluation of MINIPAR. In *Proceedings of the Workshop on Evaluation of Parsing Systems, Granada, Spain*, May 1998.
11. B. Navarro, M. Civit, M.A. Martí, R. Marcos, and B. Fernández. Syntactic, Semantic and Pragmatic Annotation in Cast3LB. In *Proceedings of Shallow Processing on Large Corpora (SproLaC), a Workshop on Corpus Linguistics, Lancaster, UK*, 2003.

12. J. Nivre, J. Hall, and J. Nilsson. Maltparser: A Data–Driven Parser–Generator for Dependency Parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy*, 2006.
13. J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. Labeled Pseudo–Projective Dependency Parsing with Support Vector Machines. In *Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA*, June 2006.
14. H. Schmid. Probabilistic Part–of–Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK*, pages 44–49, September 1994.