Procesamiento del Lenguaje Natural, núm. 43 (2009), pp. 315-323 recibido 1-05-2009; aceptado 5-06-2009

Una gramática de dependencias basada en patrones de etiquetas*

A Dependency Grammar Based on Patterns of Tags

Pablo Gamallo Otero
Dept. de Língua Espanhola
Univ. de Santiago de Compostela
pablo.gamallo@usc.es

Isaac González Sánchez
Univ. de Santiago de Compostela
osaklugo@gmail.com

Resumen: Este artículo describe un formalismo gramatical, DepPattern, concebido para escribir gramáticas de dependencias a partir de patrones de *PoS tags* enriquecidos con información léxica y morfológica. El formalismo retoma ideas de Sinclair y de la Gramática de Patrones. A partir de las gramáticas escritas con este formalismo, un compilador genera analizadores sintácticos robustos basados en expresiones regulares para 5 lenguas: español, inglés, gallego, portugués y francés. Los analizadores así generados identifican dependencias que, posteriormente, servirán para mejorar aplicaciones de PLN tales como la extracción de información tesáurica.
Palabras clave: gramática de dependencias, análisis sintáctico, extracción de información

Abstract: This paper describes a grammatical formalism, DepPattern, to write dependency grammars using Patterns of PoS tags augmented with lexical and morphological information. The formalism inherits ideas from Sinclair's work and Pattern Grammar. In addition, a compiler was implemented so as to generate robust parsers from DepPattern grammars for 5 languages: Spanish, English, Galician, Portuguese, and French. These parsers identify dependencies which can be used to improve NLP applications such as extration of similar words.
Keywords: dependency grammar, parsing, information extraction

1. Introducción

En este artículo, presentamos un formalismo basado en reglas, llamado DepPattern, pensado para que lingüistas puedan escribir fácilmente gramáticas de dependencias. Este formalismo viene acompañado de un compilador de gramáticas que genera analizadores (*parsers*) de dependencias robustos para 5 lenguas. Las principales características del formalismo son las siguientes.

En primer lugar, permite identificar dependencias entre palabras (núcleo-dependiente) mediante el uso de patrones de etiquetas morfosintácticas (PoS tags), provistas de información morfológica y léxica. Se inspira, por tanto, en la Gramática de Patrones o *Pattern Grammar* (Hunston y Francis, 1999), donde se considera que los simples patrones de etiquetas son estructuras sintácticas superficiales con capacidad para identificar estructuras más abstractas, como las dependencias o incluso el significado.

En segundo lugar, DepPattern se centra en el principio de unicidad, común a la mayoría de las gramáticas de dependencias. Según este principio, cada palabra desempeña el papel de dependiente sólo una vez. El formalimo usa el principio de unicidad de la siguiente manera. La aplicación de un patrón de etiquetas para la identificación de un par "núcleo-dependiente" puede hacer desaparecer el dependiente del espacio de busca, simplificando así el tipo de patrones necesarios para la identificación de posteriores dependencias.

En tercer lugar, nuestro formalismo plantea, como algunas teorías lingüísticas recientes, que no se puede establecer una separación tajante entre léxico y sintaxis (Sinclair, 1991; Hunston y Francis, 1999). Existen, en cualquier lenguaje natural, innumerables uni-

* Este trabajo ha sido subvencionado por la Xunta de Galicia, con cargo a los proyectos con referencia: PGIDIT07PXIB204015PR (Consellaría de Innovación e Industria) y 2008/101 (Consellaría de Educación e Ordenación Universitaria.

dades léxicas compuestas que tienen, hasta cierto punto, un comportamiento semejante a las unidades sintácticas. Por ejemplo, la expresión "tener en cuenta" es una unidad léxica de tipo verbal próxima en significado a verbos como "considerar". "valorar", "sopesar", etc, pero al mismo tiempo, es una unidad discontinua que permite la inserción de diversos complementos: "tuvo su opinión irremediablemente en cuenta". La identificación de este tipo de unidades léxicas discontinuas presupone, por tanto, el uso de los mismos mecanismos y reglas que se necesitan para identificar las unidades sintácticas.

Por último, y en referencia al análisis automático, hemos desarrollado, con licencia GPL, un compilador del formalismo DepPattern que genera analizadores robustos, escritos en Perl y basados fundamentalmente en expresiones regulares, para 5 lenguas. [1] Los analizadores generados por el compilador toman como entrada texto etiquetado y desambiguado por dos herramientas: Freeling (Carreras et al., 2004) y Tree-Tagger (Schmid, 1994). En concreto, fue definido un conversor que elabora, a partir de diferentes *tagsets* usados por Freeling y Treetagger, un conjunto común de etiquetas reconocido por los analizadores. Por el momento, el conversor tiene la capacidad de generar un tagset común a partir de 8 tagsets existentes: 3 de Freeling (español, inglés y gallego) y 5 de Treetagger (español, inglés, gallego, francés y portugués). El tagset común se definió tomando en cuenta los tagsets usados por el sistema Freeling para español y gallego, a su vez inspirados en la propuesta del grupo EAGLES, ya que son los que incorporan más información morfológica.

Para evaluar la calidad de los análisis realizados por los analizadores DepPattern, nos hemos inclinado por una evaluación indirecta a partir de una aplicación posible del análisis de dependencias, concretamente la extracción automática de información tesáurica. Para ello, hemos adaptado uno de los formatos de salida de los analizadores, de tal manera que pueda servir de entrada de un sistema de adquisición automática de tesaurus. La evaluación directa de los resultados del sistema de adquisición automática nos ayudará a evaluar indirectamente la calidad de la información sintáctica generada por los analizadores.

[1] Disponible en:
http://gramatica.usc.es/pln/tools/DepPattern.htm

Siempre que no haya ambigüedad, utilizaremos el término DepPattern, no sólo para referirnos al formalismo, sino también a los analizadores generados por el compilador de gramáticas.

El resto del artículo se organiza como sigue. Comenzaremos por abordar los fundamentos lingüísticos del formalismo (sección 2) y los trabajos relacionados (sección 3). Seguiremos con una breve descripción del mismo (sección 4), y acabaremos describiendo algunos experimentos y evaluaciones realizados (sección 5).

2. Ideas lingüísticas sobre las que se basa el formalismo

Para definir el formalismo, hemos tomado en cuenta nociones de varios enfoques lingüísticos. En concreto, nos ha interesado el trabajo de Jonh Sinclair, la teoría de la Gramática de Patrones (Pattern Grammar), así como aspectos básicos de las gramáticas de dependencias. En esta sección, presentaremos únicamente los dos primeros trabajos, por ser menos conocidos en el ámbito del PLN.

2.1. La lingüística de corpus de Sinclair

Sinclair argumenta que hay dos maneras diferentes de interpretar las expresiones lingüísticas. Por un lado, el significado de una expresión compuesta es el resultado de varias "elecciones libres" (*open choices*), realizadas de acuerdo con principios regulares de composicionalidad semántica. A esto le llama "modelo de la libre elección", y lo define como sigue (Sinclair, 1991) (pages 109-110):

> It is often called a "slot-and-filled" model, envisaging texts as a series of slots which have to be filled from a lexicon which satisfies local constraints. At each slot, virtually any word can occur. [...] All grammars are constructed on the open-choice principle.

Por otro lado, en muchos casos el significado de una expresión compuesta no es composicional, esto es, no se puede derivar directamente del de sus partes. Se trata de estructuras, en su mayoría, semi-fijadas, que sólo pueden ser interpretadas recurriendo a lo que Sinclair llama "principio idiomático" y que define de esta forma (Sinclair, 1991) (page 110):

> The principle of idiom is that a language user has available to him or her a large number of semi-preconstructed phrases that constitute single choices, even though they might appear to be analyzable into segments.

De acuerdo con Sinclair, las exprexiones semi-preconstruidas o semi-fijadas son la regla y no la excepción en cualquier lengua. Por tanto, el principio idiomático debería incorporarse a la organización de cualquier (léxico)-gramática, junto con el bien conocido modelo de la elección libre. La principal dificultad de las expresiones idiomáticas semi-fijadas es que, por una lado, son unidades con un comportamiento léxico-semántico próximo al de cualquier entrada de diccionario y, por otro, comparten propiedades sintácticas de cualquier expresión compuesta analizada mediante el modelo de la libre elección, esto es, se componen de elementos no necesariamente contiguos: *tomar [algo] en cuenta*.

Para analizar este tipo de expresiones semi-fijadas, el formalismo DepPattern permite la definición de reglas gramaticales especializadas en la identificación de unidades léxicas discontinuas y sintácticamente variables. Estas reglas se comportan como reglas sintácticas estándar, pero en vez de generar unicamente una representación sintáctica, construyen una unidad léxica que puede ser núcleo o dependiente de otras palabras en sucesivas reglas. De hecho, nuestro formalismo distingue dos tipos de reglas: reglas sintácticas que siguen el modelo de la libre elección y reglas léxico-sintácticas que respetan el principio idiomático.

2.2. Gramática de Patrones

La Gramática de Patrones, tal como se describe en Hunston y Francis (1999), puede ser vista como una "formalización de las ideas de Sinclair" (Teubert, 2007). La noción de base de esta gramática es la de patrón (*pattern*). Un patrón específico de una palabra es una organización léxico-morfo-sintáctica que agrupa etiquetas morfo-sintácticas, información léxica e información morfológica, que contribuye a seleccionar un aspecto del significado de la palabra en cuestión. Veamos algunos ejemplos de patrones:

V *a* n n	*Pedro le escribió a María una carta*
V que-subj	*Manuel sugerió que me fuera*
V inf *a* n	*Mi hermano dejó salir al cuidador*
N *sobre* n	*Una noticia sobre los negocios de mi padre*
ADJ de-inf	*El homenaje fue difícil de justificar*

A la izquierda, aparecen los patrones y a la derecha ejemplos que los instancian. En cuanto a la notación utilizada, **v** representa una frase verbal, **n**, una frase nominal, **adj**, una frase adjetiva, **que-subj** es una cláusula con verbo en subjuntivo e introducida por *que*, **inf** es una cláusula con el verbo en infinitivo, ***a*** y ***sobre*** son unidades léxicas especificas. Las mayúsculas **V** (or **N**, **ADJ**) representan las etiquetas morfo-sintácticas (PoS tags) de las palabras para las que se definen los patrones (y que aparecen subrayadas en los ejemplos). Los patrones son estructuras sintácticas de superficie que permiten describir la gramática de cualquier lengua. Las descripciones llevadas a cabo de esta guisa son menos abstractas, más léxicas y más de superficie que la mayoría de descripciones profundas efectuadas por el resto de teorías sintácticas. Para definir un patrón apenas se requieren *PoS tags* junto con alguna información léxica y morfológica. No es necesario tomar en consideración información gramatical relativa a los constituyentes y a las funciones. Se asume que esta estructura de superficie es suficiente para llevar a cabo la interpretación semántica, es decir, hay una asociación directa entre patrones superficiales y significado, sin mediación de otros niveles gramaticales de organización.

Sin embargo, bajo nuestro punto de vista, los patrones definidos por Hunston y Francis no son meras representaciones sintácticas de superficie, ya que también integran de manera implícta información gramatical de niveles superiores. En concreto, los patrones presentados arriba contienen información sobre las dependencias entre un núcleo y sus modificadores (o palabras dependientes). Tomemos como ejemplo el patrón **N *sobre* n**. Según Hunston y Francis, esta estructura debería servir para identificar aquellos casos en que el complemento ***sobre* n** depende del núcleo nominal **N**. Por lo tanto, no debe usarse pa-

ra dar cuenta de expresiones tales como *la hoja sobre la mesa*, incluidas en frases como *poner la hoja sobre la mesa*. En este caso, el núcleo del complemento preposicional ***sobre*** **n** es el verbo *poner*. De todo esto se deduce que el patrón **N** ***sobre*** **n** incluye información estructural que va más allá de la simple concatenación de PoS tags y unidades léxicas. Su correcta aplicación presupone la identificación de una depedencia sintáctica de orden superior entre el primer substantivo **N** y el núcleo del sintagma nominal **n** precedido por la preposición ***sobre***.

Al igual que la Gramática de Patrones, el formalismo DepPattern fundamenta el análisis lingüístico en el uso de cadenas de PoS tags enriquecidos con atributos morfológicos e información léxica. No obstante, en lugar de servir para identificar aspectos del significado de las palabras, nuestro propósito es identificar y generar las relaciones de dependencia inherentes a esas cadenas. Este objetivo coincide con lo expuesto por Teubert (2007), quien plantea la necesidad de enriquecer la estructura de superficie de los patrones con información lingüística más abstracta tal como las dependencias sintácticas.

3. Trabajo relacionado

Es cada vez más frecuente encontrar analizadores de dependencias de código abierto y basados en reglas. Con estas características y para el español, junto a DepPattern, podemos citar a TXALA (Atserias, Comelles, y Mayor, 2005; Carrera et al., 2008), DILUCT (Hiram Calvo, 2006) y VISL (Bick, 2006). TXALA compartte con DepPattern el uso del desambiguador morfosintáctico de Freeling. De hecho, el parser TXALA es la última etapa de la cadena de procesos de la herramienta Freeling.

Para comparar estos cuatro analizadores, veamos como se comportan con respecto a las siguientes propiedades: robustez, naturaleza de las gramáticas y licencias de uso. En cuanto a la robustez, VISL, DILUCT y DepPattern son analizadores robustos, lo que facilita su uso en tareas de extracción de información a partir de grandes cantidades de texto. Con respecto a la naturaleza de las gramáticas, VISL y DepPattern ofrecen un formalismo amigable para lingüistas, documentado en tutoriales, e inspirado en teorías lingüísticas. El formalismo de VISL se basa en la *Constraint Grammar*, mientras que el de DepPattern sigue algunos principios básicos de la Gramática de Patrones (*Pattern Grammar*). Ambos formalismos tienen un compilador que traduce las gramáticas en analizadores. Por otro lado, el formalismo gramatical de TXALA es el único de los cuatro que no se basa estrictamente en las dependencias, sino que construye primero árboles de constituyentes para después transformar estos árboles, mediante la identificación de los núcleos, en dependencias. Finalmente, en cuanto a las licencias de uso y distribución, completamente libres son TXALA, DILUCT Y DepPattern, ya que VISL no libera las gramáticas ni los parsers generados, apenas el compilador.

Por consiguiente, DepPattern es un sistema con las siguientes propiedades: tiene un analizador robusto, un formalismo gramatical documentado, compilable y estrictamente basado en dependencias, y libera tanto el compilador como las gramáticas.

4. Breve introducción al formalismo de gramáticas

DepPattern is una gramática formal, basada en reglas dependientes del contexto, que busca identificar la estructura de dependencias de las oraciones. En esta sección, introduciremos brevemente algunas de las principales características del formalismo. Para una exposición más detallada, se puede consultar el tutorial.[2]

4.1. Descripción básica de las reglas

Una gramática escrita con DepPattern consta de un conjunto de reglas dependientes del contexto. Cada regla tiene como objetivo identificar una relación *núcleo-dependiente* por medio de patrones de etiquetas morfosintácticas asociadas a información morfológica y léxica. Una regla consta de dos elementos:

- un patrón de etiquetas
- el nombre de una relación núcleo-dependiente

Veamos un ejemplo:

DobjR: VERB [DT]? [ADJ]* NOUN

[2]Disponble en:
htpp://gramatica.usc.es/pln/tools/tutorialGrammar.pdf

Los dos puntos separan el patrón de etiquetas (a la derecha) del nombre de la dependencia, DobjR, que se pretende buscar dentro del patrón. Tanto el nombre de las dependencias como el de las etiquetas morfo-sintácticas se declaran en ficheros aparte: dependencies.conf y tagset.conf, respectivamente. El nombre por omisión de las etiquetas es el que viene establecido por el conversor de etiquetas, que transforma los tagsets de Freeling y Treetagger en un tagset común. Esos nombres se pueden modificar a gusto del lingüista en el fichero tagset.conf. En cuanto a los nombres de dependencias, el lingüista tiene la posibilidad de enumerar los que desee, pero con la obligación de declarar para cada uno el tipo al que pertenece. Así en el fichero dependencies.conf aparece la línea:

DobjR HeadDep

Básicamente, DepPattern distingue 2 grandes tipos de dependencias, *DepHead* y *HeadDep*, en función de la posición del dependiente con respecto al núcleo. Las dependencias de tipo DepHead definen relaciones donde el dependiente aparece a la izquierda del núcleo, mientras que las de tipo HeadDep representan a todas aquellas donde el dependiente aparece a la derecha. En nuestro ejemplo, DobjR fue declarada como una dependencia de tipo HeadDep, por consiguiente, puede usarse para identificar los objetos directos que aparecen a la derecha de los núcleos verbales ("Dobj" es una abreviatura de Direct Object, y "R" de Right). En la regla expuesta arriba, DobjR permite identificar una relación de dependencia entre las etiquetas sin corchetes: VERB y NOUN. Dado que esta dependencia es de tipo HeadDep, sabemos que VERB es el núcleo y NOUN el dependiente. El resto de etiquetas aparecen entre corchetes ya que representan el contexto de la relación. En concreto, [DT]? significa que pueden aparecer ninguno o 1 determinantes, y [ADJ]* ninguno o varios adjetivos, todos entre el verbo y el nombre.

4.2. Extensiones del formalismo

Lo que hemos definido hasta ahora son los elementos esenciales de una regla. No obstante, es posible especificar y detallar la información de las reglas mediante el uso de estructuras de tipo atributo-valor, provistas de información morfológica y léxica, así como de operaciones de diferente naturaleza: concordancia, recursividad, herencia de atributos, modificación de valores de atributos e inclusión de nuevos pares atributo-valor. También es posible definir clases de palabras mediante listas declaradas en ficheros externos. Por último, fueron definidos operadores booleanos para las etiquetas, atributos y valores. Como describir todas las potencialidades del formalismo no es el objetivo de este artículo, nos centraremos en los nuevos elementos que aparecen en las dos reglas siguientes:

```
AdjnL: ADV<type:Q> ADJ | ADV
%
AdjnL: ADJ NOUN
Agreement: number, genre
%
```

Ambas reglas contienen la dependencia AdjnL (i.e., Adjunto a la izquierda), que fue declarada como siendo de tipo DepHead, es decir, el dependiente está a la izquierda del núcleo. En la primera regla, el atributo-valor <type:Q> es una condición que elabora la información de la etiqueta ADV. El valor "Q" se refiere a adverbios cuantificadores como *muy* o *bastante*. Esta regla, por lo tanto, identifica la dependencia de adjunción entre un adverbio cuantificador y su núcleo, que puede ser un adjetivo o un adverbio. La barra vertical es un operador de disyunción. En la segunda regla, la dependencia AdjnL identifica la relación entre un adjetivo y su núcleo nominal. "Agreement" es la operación de concordancia, y "number, genre" el nombre de los atributos cuyos valores deben compartir el núcleo y el dependiente para que la relación sea identificada.

4.3. El principio de unicidad

La mayoría de las gramáticas de dependencias presuponen el "principio de unicidad" (*uniqueness principle*). Este principio establece que cada palabra sólo tiene un núcleo, es decir, cada palabra desempeña el papel de dependiente apenas una vez. Las reglas de nuestro formalismo se aplican tomando en cuenta la unicidad del dependiente. De esta manera, una regla no sólo identifica una dependencia *núcleo-dependiente* entre dos palabras, sino que también elimina la palabra dependiente de la cadena de entrada utilizada por las siguientes reglas

que serán aplicadas. La eliminación del elemento dependiente en cada aplicación de una regla permite reducir la complejidad del espacio de busca en la definición de nuevas reglas. Veamos un ejemplo. Supongamos que escribimos una gramática con estas dos reglas:

```
SpecL: DT NOUN
%
AdjnL: ADJ NOUN
%
```

Estas reglas sirven para analizar secuencias de entrada (enunciados acabados en un punto), tales como:

```
una_DT hermosa_ADJ montaña_NOUN ._Fp
```

Para simplificar, omitimos toda la información morfológica y léxica de cada una de las palabras etiquetadas. En un primer momento, la única regla aplicable es AdjnL, que identifica la dependencia entre el adjetivo (dependiente) y el nombre (núcleo), al mismo tiempo que elimina el adjetivo de la secuencia etiquetada. Esta eliminación da lugar a una nueva secuencia:

una_DT montaña_NOUN ._Fp

que permite ahora aplicar la regla SpecL (especificador a la izquierda), la cual identifica la dependencia entre el determinante (dependiente) y el nombre (núcleo), al mismo tiempo que elimina el determinante de la secuencia de entrada. El hecho de eliminar las palabras dependientes de la secuencia de entrada permite, de forma sistemática, reducir el espacio de busca de las siguientes reglas aplicables y, así de este modo, simplificar su definición.

El análisis de un enunciado finaliza cuando no se hayen más reglas aplicables. Como resultado del análisis, el parser genera la representanción en forma de triplets de las dos dependencias identificadas:

```
(SpecL; montaña_NOUN_2; un_DT_0)
(AdjnL; montaña_NOUN_2; hermoso_ADJ_1)
```

Cada triplet se compone del nombre de la dependencia, el núcleo y el dependiente. A modo de simplificación, para cada palabra (núcleo o dependiente), sólo representamos aquí el lema, el PoS tag asociado y su posición en la frase (dejamos fuera el token y los atributos morfológicos). Existen, sin embargo, fenómenos lingüísticos, para los cuales el principio de unicidad parece demasiado riguroso (Hudson, 1990). Es el caso, por ejemplo, de los adjetivos predicativos, que tienen dos núcleos: dependen al mismo tiempo de un verbo y de un nombre sujeto o complemento directo. Para dar cuenta de estos casos y de otras situaciones análogas, el formalismo permite poner en suspenso, tanto globalmente como localmente en bloques declarativos de reglas, la eliminación del dependiente.

Así mismo, estos bloques de reglas puramente declarativos, sin eliminación del dependiente, permiten también tratar los fenómenos de ambigüedad sintáctica.

En resumen, cada regla elimina por defecto la palabra dependiente, pero existen varias opciones para no eliminarla o dejar su eliminación en suspenso, y con ello dar cuenta de los casos de no-unicidad y de ambigüedad.

4.4. Dependencias de libre elección e idiomáticas

Hasta ahora, hemos definido dos tipos de dependencias en función de la posición relativa del núcleo y el dependiente: HeadDep y DepHead. Estos dos tipos definen dependencias estándar de libre elección. Para dar cuenta del principio idiomático definido en la sección 2, nuestro formalismo introduce dos nuevos tipos de dependencias: HeadDep_lex y DepHead_lex, con los que se pueden definir dependencias que, no sólo identifican relaciones sintácticas de tipo núcleo-dependiente, sino que también construyen unidades léxicas compuestas potencialmente discontinuas.

Veamos un ejemplo. Definamos *tomar@en@cuenta* como una unidad léxica compuesta que puede o no aparecer de forma discontinua en expresiones como: “tomar tu decisión hoy en cuenta”, “tomar tu decisión en cuenta hoy”, “tomar hoy en cuenta tu decisión”, o “tomar en cuenta hoy tu decisión”. En todas estas expresiones, el verbo *tomar@en@cuenta* se combina con un objeto directo, el nombre *decisión*, y con un adjunto, el adverbio *hoy*, que pueden aparecer entre el verbo *tomar* y el complemento *en cuenta*. Existen muchas maneras de definir un conjunto de reglas que den cuenta de todas estas expresiones. Una manera sería la que muestra el cuadro 1:

La primera regla es un bloque de dos

```
TermR_lex: [VERB<lemma:tomar>] [ADV]* [NOUN]? [ADV]* PRP<lemma:en> NOUN<lemma:cuenta>
NEXT
ComplR_lex: VERB<lemma:tomar> [ADV]* [NOUN]? [ADV]* PRP<lemma:en> [NOUN<lemma:cuenta>]
%
AdjnR: VERB ADV
%
DobjR: VERB NOUN
%
SpecL: DT NOUN
%
```

Cuadro 1: Muestra de una gramática que incluye la expresión idiomática "tomar en cuenta"

subreglas (separadas por NEXT) en los que se identifican dos dependencias de tipo idiomático: la relación TermR_lex entre *en* y *cuenta* después del verbo *tomar* y la relación ComplR_lex entre *tomar* y *en* precediendo a *cuenta*. Ambas contienen elementos contextuales opcionales. Dentro del bloque, no opera el principio de unicidad y, por lo tanto, no se elimina el elemento dependiente. Además, al aplicarse reglas de tipo idiomático (marcadas por el sufijo "_lex"), el resultado final es la identificación de una unidad léxica: *tomar@en@cuenta*. El resto de reglas del ejemplo son dependencias sintácticas de libre elección. Con esta gramática, el análisis final de, por ejemplo, "tomar tu decisión hoy en cuenta" da lugar a la siguiente lista de triplets:

(AdjnR; tomar@en@cuenta_VERB_0; hoy_ADV_1)
(SpecL; decisión_NOUN_3; tu_DT_2)
(Dobj; tomar@en@cuenta_VERB_0; decisión_NOUN_3)
(ComplR_lex; tomar_VERB_0; en_PRP_4)
(TermR_lex; en_PRP_4; cuenta_NOUN_5)

Los triplets identificados se ordenan por la posición del dependiente y no por el orden de aplicación de las reglas. La primera regla aplicable es SpecL, que identifica la dependencia entre *tu* y *decisión*, al mismo tiempo que elimina el determinante de la secuencia de entrada de las siguientes reglas aplicables. Esta eliminación permite aplicar a continuación el bloque de reglas idiomáticas que genera la unidad léxica *tomar@en@cuenta*. Este bloque también identifica las dos dependencias internas de la unidad léxica: la relación TermR_lex entre *en* y *cuenta*, y ComplR_lex entre *tomar* y *en*. Seguidamente, se aplica DobjR, regla que conecta el verbo *tomar@en@cuenta* con el sustantivo *decisión*. El nombre se elimina de la entrada de la siguiente regla. Finalmente, se aplica la regla AdjnR con la que se identifica la relación entre *tomar@en@cuenta* y el adverbio *ayer*, al mismo tiempo que se elimina este último del espacio de busca. Con la eliminación sucesiva de todos los dependientes de la secuencia de entrada, llegamos al estadio en que no hay ninguna regla aplicable, ya que la secuencia que queda sin eliminar es el verbo *tomar@en@cuenta*, cabeza principal que no depende de ninguna expresión.

El formalismo permite definir clases de palabras mediante listas declaradas en ficheros externos. En el ejemplo que nos ocupa, podríamos definir una pequeña clase de verbos con *tomar* y *tener* y una de nombres con *cuenta* y *consideración*. Esta generalización nos permitiría escribir reglas para identificar expresiones idiomáticas con el mismo comportamiento sintáctico: *tomar@en@cuenta*, *tener@en@cuenta*, *tomar@en@consideración*, etc. El uso de clases de palabras también se puede extender, obviamente, a la identificación de patrones de subcategorización por medio de reglas de libre elección.

5. Extracción de información a partir de dependencias

En esta sección evaluaremos indirectamente dos analizadores DepPattern, uno para el inglés y otro para el español, por medio de su comportamiento en una aplicación concreta: la extracción automática de tesauros, donde a cada palabra de un corpus se le asocia una lista ordenada de palabras semánticamente relacionadas. Compararemos la precisión de la extracción tesáurica basada en dependencias sintácticas (Gamallo, Agustini, y Lopes, 2005) con las precisiones de dos métodos *baseline* basados en simples coocurrencias sin información sintáctica. Este experimento nos permitirá comprobar si las dependencias identificadas por los analizadores DepPattern permiten mejorar los resultados de la extracción. Tal como sugiere Kilgarriff (2003), consideramos que la evaluación indirecta de una herramienta (un analizador,

por ejemplo), observando cómo funciona en una aplicación de PLN (e.g., la extracción de tesaurus), es una alternativa válida de evaluación de la herramienta.

5.1. Los corpus

Los experimentos se realizaron sobre un corpus inglés y un corpus español. El corpus inglés es el BNC[3], etiquetado con TreeTagger. Para realizar la evaluación, seleccionamos únicamente los 10,000 nombres más frecuentes del corpus. El corpus español se construyó recuperando noticias de La Voz de Galicia y El Correo Gallego de 2006, formando una colección de artículos de 15 millones de palabras. La etiquetación morfo-sintáctica se realizó con Freeling. Fueron seleccionados para la evaluación los 15,000 nombres más frecuentes del corpus.

5.2. La evaluación

Evaluamos 3 estrategias diferentes. La primera se basa en una matriz de co-ocurrencias definidas por medio de "ventanas" de palabras del tamaño de la oración. Una oración es una cadena de palabras entre dos puntos. La segunda estrategia identifica las co-ocurrencias dentro de ventanas más reducidas (tamaño 2) y tomando en cuenta el orden de las palabras. En ambos casos se eliminan las palabras funcionales. La tercera estrategia, basada en los resultados del análisis de DepPattern, define la matriz de co-ocurrencias por medio de dependencias sintácticas. Las dos gramáticas que dieron lugar a los analizadores son muy pequeñas: contienen alrededor de 20 reglas cada una y no incluyen, por ahora, reglas idiomáticas.

Dada una estrategia y un corpus, construimos el tesauros correspondiente de la siguiente manera. Cada nombre evaluable se asocia a una lista ordenada con los 10 nombres más similares, de acuerdo con un coeficiente de similaridad. En los experimentos, hemos utilizado *cosine* y *jaccard*.

Para evaluar la calidad de la extracción tesáurica, hemos usado como referencia (*gold standard*) las asociaciones tesáuricas de WordNet (Fellbaum, 1998) para el inglés y las asociaciones de sinonimia del OpenThesaurus para el español [4]. La evaluación automática consiste en medir la calidad (en términos de *precisión*) de los 10 mejores candidatos asociados a cada nombre evaluado. En concreto, dado un nombre evaluado y sus 10 candidatos, comprobamos si éstos están semánticamente asociados al primero en el *gold standard*. La precisión se define como el número de asociaciones extraídas que también aparecen en el gold standard, dividido por el número total de candidatos, tomando en cuenta el ranking inverso.

5.3. Resultados

Métodos	corpus inglés		corpus español	
	Cosine	Jaccard	Cosine	Jaccard
Ventana (oración)	8,74	8,11	1,90	2,34
Ventana (2+orden)	11,50	10,14	1,89	3,99
DepPattern	15,18	12,97	4,94	5,18

Cuadro 2: Resultados de tres métodos de extracción sobre dos corpus

El cuadro 2 muestra la precisión obtenida (en porcentaje) por cada uno de los 3 métodos evaluados en cada corpus. Los mejores resultados fueron conseguidos por el método sintáctico en ambos corpus (valores en negrita), lo que prueba que nuestro formalismo gramatical genera analizadores basados en dependencias capaces de ayudar a mejorar la extracción tesáurica.

6. Conclusiones

En este artículo, hemos descrito algunas propiedades de un formalismo gramatical, basado en patrones de PoS tags, con el que se generan analizadores de dependencias. El formalismo, inspirado en la Gramática de Patrones, permite dar cuenta de la distinción entre reglas de libre elección e idiomáticas, al mismo tiempo que modula la aplicación o no del principio de unicidad, asunto que crea cierta polémica en el marco de las gramáticas de dependencias. Los analizadores generados por el compilador de gramáticas producen un formato de salida, con triplets de dependencias, fácilmente adaptable para su uso en sistemas de extracción de información. Nuestro objetivo a medio plazo es mejorar y actualizar las gramáticas con reglas útiles para la extracción tesáurica. En concreto, la gramáticas podrán incluir nuevas reglas siempre y cuando tal inclusión consiga mejorar los resultados de la extracción sobre un mismo corpus.

[3] http://www.natcorp.ox.ac.uk
[4] http://openthes-es.berlios.de

Bibliografía

Atserias, J., E. Comelles, y A. Mayor. 2005. Txala un analizador libre de dependencias para el castellano. *Procesamiento del Lenguaje Natural*, 35:455–456.

Bick, Eckhard. 2006. A constraint grammar-based parser for spanish. En *4th Workshop on Information and Human Technology*.

Carrera, J., I. Castellón, M. Lloberes, L. Padró, y N. Tincova. 2008. Dependency grammar in freeling. *Procesamiento del Lenguaje Natural*, 41:21–28.

Carreras, X., I. Chao, L. Padró, y M. Padró. 2004. An open-source suite of language analyzers. En *4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Fellbaum, C. 1998. A semantic network of english: The mother of all wordnets. *Computer and the Humanities*, 32:209–220.

Gamallo, Pablo, Alexandre Agustini, y Gabriel Lopes. 2005. Clustering syntactic positions with similar semantic requirements. *Computational Linguistics*, 31(1):107–146.

Hiram Calvo, Alexander F. Gelbukh. 2006. Diluct: An open-source spanish dependency parser based on rules, heuristics, and selectional preferences. En *NLDB*, páginas 164–175.

Hudson, R. 1990. *English Word Grammar*. Basil Blackwell.

Hunston, S. y G. Francis. 1999. *Pattern Grammar*. John Benjamins, Amnsterdam.

Kilgarriff, Adam. 2003. Thesauruses for natural language processing. En *Natural Language Processing and Knowledge Engineering (NLPKE)*, Beijing, China.

Schmid, H. 1994. Probabilistic part-of-speech tagging using decision trees. En *International Conference on New Methods in Language Processing*.

Sinclair, J. 1991. *Corpus, Concordance, Collocation*. Oxford University Press, Oxford.

Teubert, W. 2007. Synclair, pattern grammar and the question of hatred. *International Journal of Corpus Linguistics*, 12(2):223–248.