Análisis de dependencias no supervisado basado en Gramáticas Biléxicas

Martín A. Domínguez[1], Gabriel Infante-Lopez[1,2], and Franco M. Luque[1]

[1] Grupo de Procesamiento de Lenguaje Natural
Universidad Nacional de Córdoba - Argentina
mdoming|gabriel|lq@famaf.unc.edu.ar
[2] Consejo Nacional de Investigaciones Científicas y Técnicas

Resumen. En el presente trabajo introducimos un analizador de dependencias no supervisado basado en gramáticas biléxicas. Nuestro modelo está basado en la inferencia de una gramática biléxica, la cual se construye a partir de un conjunto de autómatas que modelan el lenguaje de dependencias para cada categoría sintáctica (verbos, sustantivos, etc). Los resultados obtenidos por nuestro analizador son comparables con el estado del arte para el idioma inglés en oraciones de hasta diez palabras. La arquitectura definida, al estar basada en autómatas, permite mucha flexibilidad; permitiendo definir distintas estructuras, dependiendo de los idiomas y de las categorías sintácticas. Debido a esto, esperamos en trabajos futuros obtener buenos resultados tanto para oraciones de mayor longitud, como para otros idiomas como el alemán, el turco y el español.

Palabras Claves: Análisis de dependencias no supervisado, Gramáticas Biléxicas, autómatas probabilísticos.

1 Introducción

El análisis de dependencias no supervisado, cuyo objetivo es adquirir conocimiento de un lenguaje a partir de un corpus sin ninguna anotación, ha tomado gran importancia en el área del procesamiento natural en los últimos años. Es utilizado como parte de sistemas de traducción automática, sistemas de diálogos, buscadores y respuesta a preguntas (question answering), entre otros. El análisis de dependencias tiene como foco construir un modelo que permita inferir el árbol de dependencias para una oración dada; a esta etapa se la denomina fase de entrenamiento. Un árbol de dependencias es un árbol cuyos nodos (internos y externos) están etiquetados usualmente con la palabra y su categoría sintáctica (POS), a su vez, la raíz es etiquetada con el símbolo ROOT. Los hijos ("dependientes") de un nodo están ordenados en una secuencia, de este modo, cada nodo puede tener un hijo a la izquierda que lo precede y un hijo derecho que lo sucede. En la figura 1 podemos ver un árbol de dependencias para la oración 297 del corpus anotado del inglés Penn TreeBank (PTB) [1].

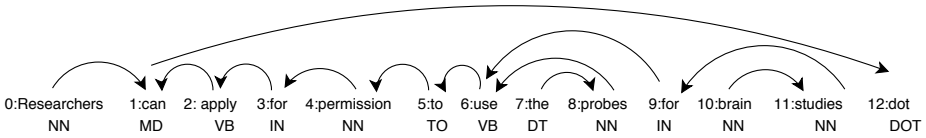

Fig. 1. Árbol de dependencias correspondientes a la oración 297 del Penn Tree Bank.

En el análisis no supervisado de dependencias, para la fase de entrenamiento contamos con un corpus de oraciones sin ninguna anotación, salvo el punto de fin de oración. Para evaluar un analizador de dependencias, usualmente se cuenta con un corpus con las dependencias anotadas de manera correcta (Gold Standard), contra el cual se comparan los árboles obtenidos por el modelo a ser evaluado. Para poder compara dos corpus de dependencias que tengan como base la el mismo conjunto de oraciones, se toma un árbol de cada corpus que corresponda a la misma oración y considerando que los arboles de dependencias son grafos dirigidos, se toman las siguientes medidas:

- Precisión dirigida: porcentaje de coincidencia en aristas dirigidas entre los árboles comparados.
- Precisión no-dirigida: porcentaje de coincidencias en aristas no dirigidas.

La tesis de Klein [2] fue el primer trabajo que obtuvo resultados de consideración, y marcó una dirección para abordar el análisis de dependencias no supervisado. En ese trabajo se presenta un modelo generativo para análisis de dependencias no supervisado, Dependency Model with Valence (DMV), que fue el primero en superar la performance del algoritmo trivial de agrupación a derecha (right branching algorithm, que asigna como dependencia a cada palabra de la oración, la palabra de la derecha).

El analizador DMV es un modelo generativo que utiliza el algoritmo de optimización EM (Expectation Maximization) [3]. En cada paso se reestiman los parámetros del modelo basado en la probabilidad de todos los posibles sub-árboles de dependencias para las oraciones del corpus de entrenamiento.

En la actualidad, los analizadores con mejor desempeño están aún basados en el modelo DMV. En [4] los autores presentan un analizador EVG (Extended Valence Grammar), que es una generalización del modelo de DMV, que alcanza una de los desempeños más altos hasta el momento. Los autores en dicho trabajo, utilizan una variante más complicada del algoritmo EM para la fase de entrenamiento y agregan algunas etapas de suavizado basados en corpus paralelos.

El resto del presente artículo se organiza de la siguiente manera: en la Sección 2 daremos una breve definición de las gramáticas biléxicas y además, describiremos cómo se puede inferir una gramática biléxica a partir de un corpus de dependencias. En la Sección 3, presentamos la arquitectura de nuestro analizador de dependencias que se utiliza en la fase de entrenamiento para obtener la gramática óptima para el análisis

de dependencias. En la Sección 4 presentamos los resultados obtenidos por nuestro analizador y los comparamos con otros analizadores en el estado del arte. En la Sección restante, se encuentran las conclusiones y algunas líneas de investigación para trabajos futuros.

2 Gramáticas Biléxicas

En la presente sección, en primer lugar, definiremos las gramáticas biléxicas, que fueron presentadas en [5]. En el mencionado trabajo los autores demuestran, además, que estas gramáticas son equivalentes a una gramática libre de contexto probabilística (PCFG). En segundo lugar, describiremos de qué manera se puede inferir una gramática biléxica a partir de un corpus de dependencias. Durante el proceso de inferencia gramatical, es necesario inducir autómatas probabilísticos. En este sentido, usamos dos métodos para inducir dichos autómatas: el algoritmo MDI de [6] y una manera ad-hoc definida por nosotros que describiremos con más detalle.

Una *Gramática Biléxica* B es una 3-upla $(R_o, \{r_c\}_{w \in C}, \{l_c\}_{c \in C},)$ donde:

- C es un conjunto de palabras o POS, C contiene un símbolo distinguido ROOT.
- para cada palabra $c \in C$, l_c y r_c son dos autómatas probabilísticos con símbolos iniciales S_{l_c} y S_{r_c} respectivamente. Cada autómata acepta un lenguaje en C^* .

En el presente trabajo consideraremos que C es el conjunto de POS. La idea intuitiva detrás de la gramática biléxica es que, por ejemplo, dado un elemento de C, entonces S_{l_c} acepta el lenguaje de las dependencias a izquierda de c y S_{r_c} acepta el lenguaje de las dependencias a derecha.

2.1 Inducción de gramáticas biléxicas

Para obtener una gramática biléxica a partir de un corpus de dependencias, extraemos de cada una de las oraciones, las dependencias a izquierda y derecha para cada POS. Con estas dependencias se construyen dos multiconjuntos para cada POS p. Por ejemplo, si el corpus de dependencias fuera el árbol de la Figura 1, en la Tabla 1 podemos ver los multiconjuntos a izquierda y derecha, extraídos de las cinco primeras palabras de la oración, para los POS: `NN`, `MD`, `VB`, `IN` y `NN`.

Una vez realizado este proceso, para cada POS p, tendremos dos multiconjuntos M^p_{left} y M^p_{right}, los cuales se utilizan como material de entrenamiento para inducir los respectivos autómatas l_p y r_p, que aceptan los lenguajes de dependencias a izquierda y derecha respectivamente de p. Una vez obtenidos los autómatas, es posible definir una gramática biléxica $B = (ROOT, \{r_p\}_{p \in POS}, \{l_p\}_{p \in POS},)$.

Word #	i	T^i_{left}	T^i_{right}
0	NN	{NN}	{NN}
1	MD	{MD NN}	{MD VB ROOT}
2	VB	{VB}	{VB IN}
3	IN	{IN}	{IN NN}
4	NN	{NN}	{NN TO}

Table 1. Multiconjuntos a izquierda y derecha extraídos de las primeras cinco palabras del árbol de dependencias de la Figura 1. En todos los casos, las instancias del lenguaje de dependencias, la primer "palabra" está compuesta por el POS considerado. Las dependencias a izquierda deben ser leidas de derecha a izquierda.

2.2 Inducción de autómatas probabilísticos

Un inductor de autómatas es un algoritmo que toma como entrada un multiconjunto de palabras que debe aceptar el autómata, el algoritmo devuelve un autómata probabilístico que generaliza al conjunto de entrenamiento. Hemos utilizado dos algoritmos para inferir autómatas.

Por un lado, uno de los inductores ya existentes, MDI (Minimum Discrimination Information), [6], en el cual en cada paso del ciclo de inferencia se busca balancear la posible mejora en el desempeño y al mismo tiempo los cambios introducidos en el autómata previamente inferido, usando la desigualdad de Kullback-Leibler como medida de la diferencia entre autómatas probabilísticos.

Por otro lado, implementamos una manera ad-hoc para inducir autómatas. La idea principal es definir una estructura fija de un estado final distinguido del cual no parte ninguna transición y N estados numerados $1 \dots N$ y transiciones entre estos con un símbolo y una probabilidad asociado a cada una, con las siguientes restricciones:

- las transiciones de un estado de $i \in 1 \dots n-1$ pueden alcanzar o bien el estado final o el estado $i+1$
- las transiciones del estado n puede alcanzar o bien el estado final o a sí mismo.

Con esta estructura y con un conjunto de entrenamiento de las palabras que debe aceptar el autómata, podemos definir un algoritmo de inducción. Durante la etapa de inducción del autómata se calculan las transiciones de cada estado. Para calcular la relación de las transiciones utilizamos una tabla con las siguientes entradas:

1. m, que representa el identificador del estado de donde parte la transición, $1 \leq m \leq N$
2. p_m , el símbolo asociado a la transición.
3. o, identificador del estado al que se llega mediante la transición, $1 \leq o \leq N$
4. c^p_m contador de ocurrencias del símbolo p para el estado m

Notemos que, por las restricciones impuestas a la estructura de los autómatas, si $m < N$ entonces $o = m + 1$, y si $m = N$ entonces debe ser que $o = N$ o, en ambos casos

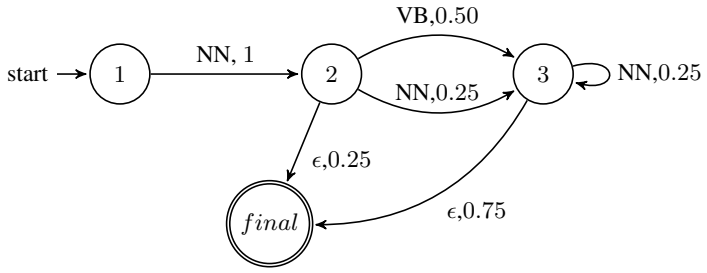

Fig. 2. Autómata probabilístico inducido con el procedimiento ad-hoc para una estructura fija de 3 estados y un corpus de 4 palabras.

se puede alcanzar el estado final. Inicialmente, los contadores se igualan a cero. En cada paso del procedimiento se lee una palabra, supongamos que el procedimiento se ha aplicado para las primeras $k-1$ palabras del material de entrenamiento, veamos cómo se actualiza la tabla de transiciones para el paso k. Sea $T_k = p_1^k \dots p_l^k$ la k-ésima palabra del material de entrenamiento, entonces la tabla de transición se modifica de la siguiente manera:

1. para j tal que $1 \leq j \leq l$ y además $j < N$ se actualizan, las siguientes entradas sumando uno a los respectivos contadores: $(j, p_j^k, j+1, ck_j^{p_j^k} + 1)$
2. para j tal que $N \leq j \leq l$ se actualiza la entrada de la tabla correspondiente al N-ésimo estado con $(N, p_j^k, N, ck_N^{p_j^k} + 1)$

donde $ck_N^{p_j^k}$ son los respectivos contadores hasta el paso de inducción $k-1$. Una vez calculada la tabla de transiciones, los contadores se normalizan con la cantidad de símbolos leídos en cada posición. Por último la probabilidad de alcanzar el estado final en un estado i es igual a la cantidad de palabras de longitud $i-1$ en el material de entrenamiento. Por ejemplo, supongamos que queremos inducir un autómata con $N = 3$ estados utilizando el siguiente material de entrenamiento compuesto de cuatro oraciones:

```
NN
NN VB
NN NN
NN VB NN
```

el autómata probabilístico resultante del proceso de inducción ad-hoc para una estructura fija de 3 estados es el de la Figura 2.

3 Arquitectura para el análisis de dependencias

En esta sección describiremos el funcionamiento de nuestro analizador de dependencias basado en gramáticas biléxicas. La arquitectura utilizada para el analizador implementa un procedimiento de optimización, que obtiene la gramática óptima para el análisis de dependencias, una vez alcanzada la convergencia del procedimiento que maximiza un estimador de máxima verosimilitud. La arquitectura del analizador puede verse en la Figura 3. En el resto de la sección describiremos cómo funciona cada componente del analizador.

– **Inicialización:**

La inicialización del procedimiento de optimización es un conjunto de árboles de dependencias aleatorios, similares a los utilizados en el analizador DMV [2]. Este banco de árboles iniciales se obtiene a partir del corpus sin anotar, utilizando una distribución de probabilidad basada en dos principios que cumplen los árboles de dependencias de los lenguajes naturales:

- En general los dependientes de una palabra se encuentran en las cercanías.
- En general, en las oraciones, cada palabra tiene a lo sumo un dependiente a cada lado.

Teniendo en cuenta estos principios, generamos los árboles con una distribución de probabilidad tal que la probabilidad de que una palabra dependa de otra es inversamente proporcional a la distancia entre ambas.
La idea es que para cada longitud de oración generamos, con tags distintos, inducimos los autómatas de cada POS utilizando el siguiente material de entrenamisnto: dalda la oración $TAG_1 \dots TAG_n$ para cada label TAG_i generamos los dos lenguajes de entrenamiento, de izquierda y derecha respectivamente: $TAG_i TAG_{i+1}$
..
$TAG_i \dots TAG_n$
$TAG_i TAG_{i-1}$
..
$TAG_i \dots TAG_1$
y posteriormente inducimos la gramática tal como se explica en la sección 2.1.

– **Inductor de Gramáticas Biléxicas:**

Con el corpus inicial de dependencias, el módulo de inducción de gramática extrae el lenguaje de las dependencias a izquierda y derecha para cada POS. Posteriormente, se inducen los autómatas correspondientes y finalmente, construye una Gramática Biléxica tal como se vio en la Sección 2.1.

– **Analizador de dependencias:**

En este módulo de la arquitectura se implementa un analizador de dependencias que obtiene los K- árboles más probables para una oración de entrada. Para implementar

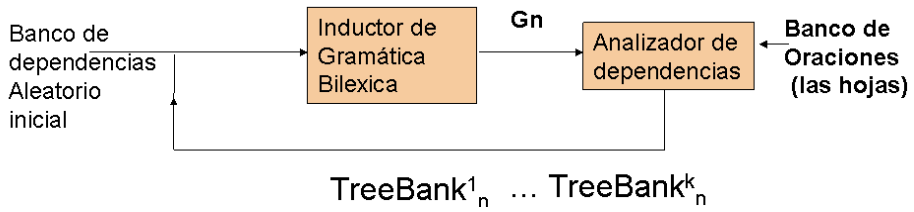

Fig. 3. Diagrama esquemático de una iteración de nuestra arquitectura para el análisis de dependencias. Partiendo de un corpus de dependencias inicial, se induce una gramática biléxica G_n, que posteriormente se utiliza para analizar un conjunto de oraciones (que son las hojas del banco inicial de dependencias). El conjunto de árboles de salida del analizador, retro alimentan la arquitectura.

el analizador, transformamos la gramática biléxica obtenida por el módulo anterior, en una gramática libre de contexto probabilística en forma normal de Chomsky, tal como puede verse en [7]. Luego, implementamos un algoritmo CYK [8] que obtiene los K-árboles más probables para cada oración de entrada.

Los K árboles obtenidos para cada oración se utilizan nuevamente como entrada de la arquitectura. Para esta retroalimentación, cada árbol se replica de manera proporcional de acuerdo a la probabilidad asignada por el analizador de dependencias.

La secuencia de interacción antes descripta constituye un paso de iteración de la arquitectura esquematizada en la Figura 3.

Como en todo método de optimización, es necesario definir una medida para maximizar o minimizar. La medida considerada es el estimador de máxima verosimilitud (maximum likelihood) . Dicha medida se calcula sobre la distribución de probabilidad de los árboles, es decir, que el módulo analizador de dependencias, calcula dicha medida promediando las probabilidades de cada uno de los árboles obtenidos. Es importante destacar que el proceso de optimización ha convergido en todo los experimentos que se han realizado, es decir, que a partir de un paso de iteración, no hay variación del estimador de máxima verosimilitud.

El procedimiento que acabamos de describir que obtiene una gramática G_n que maximiza el estimador de máxima verosimilitud, lo denominamos fase de entrenamiento.

4 Resultados experimentales

En esta sección presentaremos los resultados de los experimentos realizados con nuestro analizador de dependencias. Todos los experimentos se realizaron con categorías sintácticas (POS) en vez de palabras. Las oraciones utilizadas para las evaluaciones fueron extraídas del PTB, y sólo aquellas oraciones de a lo sumo diez palabras de longitud (usualmente se denominan WSJ10). Las secciones desde la 1 a la 22 del WSJ10 (en

total 6266 oraciones) se utilizaron para la fase de entrenamiento, es decir, para obtener la gramática óptima tal como se describió en la sección 3. Para calcular las medidas de calidad con la gramática obtenida se analizaron las oraciones de la sección 23 (un total de 299 oraciones). En la tabla 2 reportamos la precisión dirigida y la no dirigida de nuestro analizador. Además, comparamos nuestros resultados con los obtenidos por DMV y las distintas variantes que resumen el estado del arte en análisis de dependencias. Estos resultados han sido extraídos de [9], eligiendo sólo los resultados más relevantes reportados para WSJ10. En la primera fila de la Tabla 2 se reporta la precisión obtenida por el algoritmo de asignación a derecha. En la segunda fila, se muestra el desempeño obtenido por el algoritmo estándar de DMV [2]. En la tercera fila se incluye la medida de calidad obtenida por la variante de DMV con "baby steps" reportada en [9], cabe señalar que dicho trabajo tiene el mejor desempeño para oraciones de mayor longitud que no reportamos en el presente trabajo. En la cuarta fila se reportan los resultados del analizador EVG [4], el cual posee el mejor desempeño hasta el momento en oraciones de hasta longitud diez. En las dos últimas filas se muestran los resultados de nuestro analizador. En la fila cinco se puede observar la precisión dirigida de $\%61.2$ obtenida utilizando el algoritmo de MDI para inducir autómatas. En la última fila, se observa que al utilizar la inducción de autómatas ad-hoc descripta en la sección 3 alcanzamos una precisión dirigida del $\%63.4$, la cual es muy cercana a la obtenida por el analizador EVG del $\%65$.

Analizador	autores	dirigida	no dirigida
1. asignación a derecha	(Klein y Manning, 2004)	38.4	
2. DMV-standard	(Klein y Manning, 2004)	45.8	
3. DMV-babysteps	(Spitkovsky y Alshawi, 2010)	55.1	
4. EVG-Smoothed (skip-head)	(Headden y otros, 2009)	65.0	
5. este trabajo con inducción autómata MDI		61.2	68.5
6. este trabajo con inducción autómata ad-hoc		63.4	69.9

Table 2. Resultado de nuestro analizador comparado con los baselines (asignación a derecha y DMV Standard) y con analizadores en el estado del arte, como la variante "baby steps" de DMV y, por otro lado, EGV. Los resultados presentados fueron tomados de [9] .

5 Conclusiones y trabajos futuros

Hemos presentado un analizador de dependencias no supervisado, basado en gramáticas biléxicas, cuyos resultados para el idioma inglés son comparables a los resultados en el estado del arte. Cabe señalar que los analizadores de dependencias actuales con mejores

resultados son todos variaciones del analizador DMV, en contraste, nuestra arquitectura es considerablemente diferente y más flexible. Esta flexibilidad esta dada por la posibilidad de utilizar distintas estructuras de autómatas según la necesidad. Por ejemplo, se podrían definir distintas estructuras dependiendo de la categoría sintáctica y del idioma.

Por este motivo una línea de investigación futura, está centrada en estudiar distintas técnicas no supervisadas que aprendan mejores estructuras de autómatas para las distintas categorías sintácticas de cada idioma. Con esta fase, esperamos obtener un mejor desempeño en otros idiomas como español, turco, alemán, etc.

Por otro lado, otra tarea que desarrollaremos es incorporar una fase a la arquitectura que genere un Corpus inicial distinto del actual, que nos permita alcanzar mejores resultados. Además, la idea es que, para cada oración de entrada se generen también K árboles, y no uno solo como en la actualidad. De esta manera podemos atacar el problema de la posible escasez de datos en la inducción de autómatas.

En un plano más general, podemos concluir, debido a los buenos resultados obtenidos, que las gramáticas biléxicas son un formalismo que para modela el lenguaje de las dependencias alcanzando performances comparables a las del estado del arte que se basan en otros tipos de modelos más complejos. Otro trabajo donde se muestra la utilidad de usar gramáticas biléxicas para modelar el lenguaje de dependencias es [10], donde los autores las usan para obtener particiones del conjunto de POS, sobre todo en los verbos, que mejoren el desempeño de analizadores sintácticos supervisados.

References

1. Marcus, M., Santorini, B.: Building a large annotated corpus of English: The Penn treebank. Computational Linguistics **19** (1993) 313–330
2. Klein, D.: THE UNSUPERVISED LEARNING OF NATURAL LANGUAGE STRUCTURE. PhD thesis, STANFORD UNIVERSITY (2005)
3. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B **39** (1977) 1–38
4. W. P. Headden, M.J., McClosky., D.: Improving unsupervised dependency parsing with richer contexts and smoothing. In: In Proc. of NAACL-HLT. (2009)
5. Eisner, J.: Three new probabilistic models for dependency parsing. In: Proc. COLING 1996. (1996)
6. Franck Thollard, P.D., de la Higuera, C.: Probabilistic `dfa` inference using kullback-leibler divergence and minimality. In: 17th International Conference on Machine Learning. (2000)
7. Infante-Lopez, G.: Two-Level Probabilistic Grammars for Natural Language Parsing. PhD thesis, Universiteit van Amsterdam (2005)
8. Younger, D.H.: Recognition and parsing of context-free languages in time n3. In: Information and Control. (1967) 189–208
9. Valentin I. Spitkovsky, H.A., Jurafsky, D.: From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In: In Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010). (2010)

10. Domínguez, M.A., Infante-Lopez, G.: Searching for part of speech tags that improve parsing models. In: Proceedings of the 6th international conference on Advances in Natural Language Processing, GoTAL,Gotemburgo, Suecia. (2008)