Analizador sintáctico probabilístico con clasificador de argumentos de verbo para el idioma español

John Alexander Vargas

Universidad del Valle
Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Cali, Colombia
2015

Analizador sintáctico probabilístico con clasificador de argumentos de verbo para el idioma español

John Alexander Vargas

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ingeniería, Enfásis en Ingeniería de Sistemas y Computación
Plan 7721

Director:
Raúl E. Gutierrez de Piñeres Reyes Ph.D.

Línea de Investigación:
Procesamiento del Lenguaje Natural
Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle
Cali, Colombia
2015

Resumen

En este trabajo de investigación se desarrolló un clasificador de argumentos de verbo basado en aprendizaje automático usando la técnica de máquinas de soporte vectorial [13, 30], y se integró en un analizador sintáctico probabilístico para el idioma español, con dos propósitos, proveer información semántica sobre la distinción entre argumentos de verbo de un árbol sintáctico, e intentar mejorar la precisión del árbol sintáctico obtenido por el analizador sintáctico. La integración se realizó sobre la implementación de Dan Bikel [4] usando el segundo modelo de Collins [11, 12], en el que define probabilidades basadas en marcos de subcategorización, que son conjuntos de constituyentes requeridos por otros constituyentes, en este trabajo se hace una modificación al segundo modelo para que tenga en cuenta la clasificación de los argumentos requeridos por constituyentes verbales, según los resultados de la máquina de soporte vectorial. Como resultado de la investigación se logró obtener un árbol sintáctico con información adicional sobre los argumentos verbales, clasificados entre complementos y adjuntos usando como corpus de entrenamiento el Ancora [24] anotado con más de 500.000 palabras del idioma español.
Palabras clave: máquinas de aprendizaje, máquinas de soporte vectorial, analizador sintáctico probabilístico, procesamiento de lenguaje natural, argumentos de verbo.

Abstract

This research developed a classifier based on arguments of the verb based on automatic learning using the technique of vector support machines [13, 30], and was integrated into a probabilistic parser for Spanish language with two purposes, first to provide semantic information about the distinction between arguments of a verb from a syntax tree; the second purpose is to try to improve the precision of the syntactic tree obtained by the parser. To accomplish this the implementation of a probabilistic parser developed by Dan Bikel in his doctoral thesis [4] was modified. As a baseline model Bikel chose to instantiate the second model of Michael Collins [11, 12], he defined probabilities based on subcategorization frames, which are sets of constituents required by other constituents when the syntax tree is formed. At this point the arguments of the verb classifier is used to compliment the information used by the probabilistic model in predicting the syntactic structure. As a result of the investigation it was possible to obtain a syntactic tree with further information on the verbal arguments, classified between complements and adjuncts. In this research we worked with ANCORA [24] corpus annotated with over 500,000 words from the Spanish language.
Keywords: machine learning, support vector machine, probabilistic parsing, natural language processing, arguments of verb

Contenido

1 Introducción

Los analizadores sintácticos probabilísticos hoy en día son una de las técnicas del procesamiento del lenguaje natural más usadas en tareas como, renombramiento de entidades, sistemas para resúmenes, sistemas de pregunta-respuesta, social media, etc. El uso de estos analizadores está comprobado por la precisión al momento de anotar cada una de las sentencias que son testeadas. En este sentido, existen trabajos estructurados como el de Charniak [8], Bikel [4] y Collins [12] que se constituyen en pilares de los analizadores en la cual las estructuras y subestructuras de un lenguaje son extraidas de manera automática. El análisis sintáctico de una oración escrita en lenguaje natural consiste en recuperar la estructura sintáctica o árbol sintáctico asociados a esa oración. Para ello se utiliza un gramática que describe la estructura sintáctica del lenguaje. Un algoritmo determina cuál es el árbol sintáctico de la oración y en la medida que la oración no pertenezca al lenguaje no se proporciona ninguna solución. En Allen [2] se puede encontrar una descripción de los algoritmos clásicos que realizan *análisis sintáctico completo*. Estos algoritmos ofrecen muy buenos resultados para lenguajes restringuidos definidos por una gramática de cobertura limitada. Según Molina [23] el principal problema de un analizador sintáctico completo radica en seleccionar el análisis sintáctico correcto de una oración de entre todos los posibles. En tal sentido, surgen los problemas de ambigüedad estructural que son difíciles de resolver cuanto más compleja es la gramática. Sin embargo, no todas las aplicaciones de PLN requieren un análisis sintáctico completo de la oración.

Uno de los trabajos más importantes sobre analizadores probabílisticos es el de Michael Collins [12] que propusó un modelo generativo sobre gramáticas libres de contexto probabilísticas lexicalizadas para oraciones escritas en inglés. Este modelo fué extendido por Collins para hacer distinción entre complementos y adjuntos adicionando probabilidades sobre marcos de subcategorización, el cual consiste en crear conjuntos de consituyentes requeridos por otros para formar árbols sintácticos. Más adelante Dan Bikel [4] implementó una motor multi-lenguaje en java, instanciando el modelo extendido de Collins para análisis sintáctico probabilístico.

En este trabajo se realizó la integración de un analizador sintáctico probabilístico para el español que usa el modelo computacional de Collins-Bikel con un clasificador de argumentos de verbo de adjunción y complemento usando técnicas de aprendizaje automático como las máquinas de soporte vectorial (SVMs) [13, 16]. Se escogió esta técnica porque ya se han

realizado trabajos para esta tarea, como el de Pradhan & Jurafsky [30, 10] que utilizan SVMs para clasificación automática de argumentos para hacer análisis semántico superficial también para oraciones en inglés. Este clasificador sirve para la extracción y etiquetado de las características de adjunción y complemento que no están definidas en el analizador de Bikel. Hacer la distinción de argumentos de verbo entre complementos y adjuntos permite extraer del árbol sintáctico, un subárbol básico que conserva la esencia y sentido de la oración inicial. De igual manera se usó el clasificador de argumentos para estudiar la posibilidad de mejorar la presición del analizador sintáctico en el momento de la verificación de los marcos de subcategorización del segundo modelo de Collins.

El documento esta conformado en el primero capítulo por la introducción, presentando el objetivo general y objetivos específicos. El segundo capítulo muestra el marco teórico sobre análisis sintáctico probabilístico, los tres modelos porpuestos por Collins, la implementación realizada por Dan Bikel y los paquetes que fueron necesarios construir en este trabajo para su configuración al idioma español. El tercer capítulo muestra la definición de características lingüisticas usadas para el clasificador semántico de argumentos de verbo, la parametrización usada en la implementación de máquinas de soporte vectorial implementada con la librería *libsvm* [7] y la integración del clasificador en el algoritmo CYK [27]. En el cuarto capítulo se explica el procedimiento realizado para evaluar el analizador sintáctico probabilístico tomando como línea base la implementación de Bikel configurada para el español, y modificando este para usar el clasificador semántico de argumentos de verbo. El último capítulo presentan las conclusiones obtenidas en el trabajo de investigación así como recomendaciones para trabajos futuros.

1.1. Problema

El análisis sintáctico de una oración escrita en lenguaje natural consiste en recuperar la estructura sintáctica o árbol sintáctico asociados a esa oración. Para ello se utiliza un gramática que describe la estructura sintáctica del lenguaje. Un algoritmo determina cuál es el árbol sintáctico de la oración y en la medida que la oración no pertenezca al lenguaje no se proporciona ninguna solución. En Allen [3] se puede encontrar una descripción de los algoritmos clásicos que realizan *análisis sintáctico completo*. Estos algoritmos ofrecen muy buenos resultados para lenguaje restringuidos definidos por una gramática de cobertura limitada. Según Molina [23] el principal problema de un analizador sintáctico completo radica en seleccionar el análisis sintáctico correcto de una oración de entre todos los posibles. En tal sentido, surgen los problemas de ambigüedad estructural que son difíciles de resolver cuanto más compleja es la gramática. Sin embargo, no todas las aplicaciones de PLN requieren un análisis sintáctico completo de la oración. Uno de los principales problemas que se encuentran en el análisis sintáctico es la ambigüedad [9], dado que se pueden asociar varias estructuras de frase o árboles a una misma oración. Los últimos enfoques utlizados para resolver este problema

son los analizadores sintácticos probabilísticos. La mayoría de estos analizadores como el de Collins [11], Bikel [4] están aplicados al idioma inglés. Existen algunas implementaciones para el idioma español como el de Brooke Cowan [14] alcanzando una mejoría de precisión en este trabajo del 81.2 % de exactitud teniendo como línea base el analizador de Collins. Collins en su primer modelo esta parametrizado por la identificación del núcleo de la frase y una métrica de distancia, en el segundo modelo esta parametrizado con la identificación y clasificación de argumentos del núcleo, distinguendolos entre complementos y adjuntos. Para realizar esta tarea de identificación y clasificación usa un modelo basado en probabilidades y reglas. Este trabajo de invesitgación explora otras características sintácticas y aplica técnicas de máquinas de soporte vectorial para realizar dicha clasificación. Se han realizados trabajos en clasificación de argumentos conocidos como etiquetamiento de roles semánticos [33] que han mostrado resultados satisfactorios al usar máquinas de vectores de soporte. La idea es usar este clasificador como parámetro en el modelo 2 de Collins y de esta manera verificar si se puede mejorar la precisión de exactitud del analizador sintáctico probabilístico.

1.2. Objetivos

1.2.1. Objetivo General

Estudiar la posibilidad de mejorar el nivel de precisión en el análisis sintáctico probabilístico para el idioma español parametrizando el modelo de Collins en el algoritmo de Dan Bikel con las características de adjunción y complemento.

1.2.2. Objetivos Específicos

1. Definir características lingüisticas de adjunción y complemento del idioma español.
2. Implementar el algoritmo de Bikel como modelo de línea base para el análisis sintáctico de la sentencia de entrada en español.
3. Definir el conjunto de ejemplos positivos y negativos para el entrenamiento del algoritmo de aprendizaje usando las características de adjunción y complemento sobre el modelo base de Bikel.
4. Implementar los algoritmos de de extracción y clasificación para el algoritmo de aprendizaje usando las características de adjunción y complemento.
5. Implementar la parametrización del modelo de Collins en el algoritmo de Bikel, con las clasificación de complementos y adjuntos obtenidas con el modelo de SVM.
6. Implementar los algoritmos de desempeño para analizar los resultados entre el modelo baseline y el modelo modificado de Bikel.

2 Marco Teórico

2.1. Análisis Sintáctico

El análisis sintáctico (en inglés: *parsing*) es uno de los problemas fundamentales que existe en procesamiento de lenguaje natural y es necesario en diversas aplicaciones como la extracción de información, traducción de textos y reconocimiento del habla. Consiste en encontrar un algoritmo que reciba como entrada una frase escrita en lenguaje natural y retorne como salida la estructura sintáctica que esta basada en una gramática previamente establecida. Esta estructura sintáctica se encuentra representada en un árbol de estructura de frase. El árbol mostrado en la figura **2-1** [38] corresponde a la estructura de la frase "Juan vió un hombre" según la gramática de la figura **2-2**.

Para una frase dada, se pueden presentar varios árboles de estructuras sintácticas, este es el principal problema a trabajar en el análisis sintáctico conocido como ambigüedad. Para dar solución a este tipo de problemas se definen las gramáticas probabilísticas libres de contexto (PCFGs) que son entrenadas por grandes corpus de los cuales se generan las probabilidades, buscando siempre la mejor probabilidad para una sentencia de entrada. En este sentido, se han desarrollado varias soluciones basados en enfoques probabilísticos o usando técnicas de aprendizaje automático para poder predecir la estructura con la mayor precisión posible. También existen datos de entrenamiento alrededor de 50.000 sentencias con sus árboles de estructura sintáctica asociados para experimentación como el corpus Penn WSJ Treebank [1] de la universidad de Pensilvania para el caso del idioma inglés y el corpus de Ancora [24]

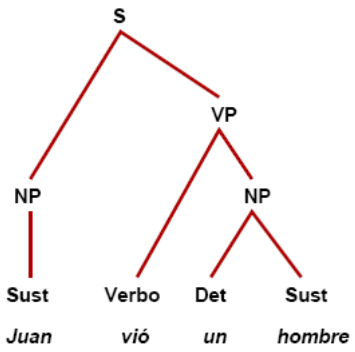

Figura **2-1**: Ejemplo de un árbol de estructura de frase [38]

S → NP VP
NP → Sust
NP → Det Sust
VP → Verbo NP

Figura **2-2**: Ejemplo de una gramática libre de contexto [38]

de la Universidad de Barcelona que cuenta con diferentes niveles de anotación [25] y es el usado en este trabajo de investigación.

2.1.1. Gramáticas Libres de Contexto (CFG: Context Free Grammar)

Formalmente una gramática libre de contexto [12] es una tupla $G = (N, \Sigma, P, S)$, donde

- N es un conjunto finito de símbolo no terminales.
- Σ es un conjunto finito de símbolos terminales.
- P es un conjunto de producciones o reglas. Una regla P toma la forma $\alpha \rightarrow \beta$ donde $\alpha \in N$ y $\beta \in \{N \cup \Sigma\}*$.
- $S \in N$ es el símbolo de inicio.

Una gramática define un lenguaje formal y se usa para obtener una estructura gramátical de frases. Es una idea formalizada por Chomsky en el año 1957 [9]. Las gramáticas capturan la noción de sintagma y de orden. La motivación original para las gramáticas libres de contexto fue la descripción de lenguajes tal como ocurre con las siguientes reglas:

⟨oración⟩ → ⟨frase sustantivo⟩ ⟨ frase verbal⟩
⟨frase sustantivo⟩ → ⟨determinante⟩ ⟨sustantivo⟩
⟨determinante⟩ → el
⟨sustantivo⟩ → niño
⟨frase verbal⟩ → juega

Donde los no terminales están escritos dentro de corchetes angulares y los terminales son en este caso las palabras “el”, “niño” y “juega”.

2.1.2. Gramáticas de Adjunción árboles (TAG)

A continuación se describen las gramáticas de adjunción de árboles basado en la definición hecha por Miguel Prado [38] en su tesis doctoral.

Joshi, Levy y Takashi [17] definieron las gramáticas de adjunción de árboles como una extensión de las gramáticas libres de contexto basado en reglas para escribir los nodos de los ároles como otros árboles [21, 37].

Formalmente, una gramática de adjunción de árboles se define como una quíntupla $(V_T; V_N; I; A; S)$ donde:

- V_T es un conjunto finito de símbolos terminales
- V_N es un conjunto finito de símbolos no-terminales. Donde $V_T \cap V_N = \emptyset$.
- I es un conjunto finito de árboles iniciales.
- A es un conjunto finito de árboles auxiliares.
- $S \in V_N$ es el axioma de la gramática.

Para las gramáticas de adjunción de árboles se deben tener en cuenta las siguientes definiciones:

- **Árboles iniciales:** Son árboles que tienen la raíz etiquetada por el axioma de la gramática $S \in V_N$.
- **Árboles auxiliares:** Son árboles iniciales con la excepción de que la etiqueta de su raíz puede ser un no-terminal arbitrario y porque uno de sus nodos hoja, que recibe el nombre de *pie* esta etiquetado por el mismo no-terminal que etiqueta su raíz.
- **Árboles elementales:** Corresponden a los árboles iniciales y auxuliares.
- **Espina:** Es el camino desde el nodo raíz hasta el nodo pie.
- **Árboles derivados:** Son árboles creados por la combinación de los árboles elementales, los cuales a su vez se pueden combinar con otros árboles para formar árboles derivados más grandes.

Los árboles se combinan mediante una operación denominada *adjunción* , la cual se muestra gráficamente en la figura **2-3**. Mediante esta operación de adjunción se construye un nuevo árbol a partir de un árbol auxiliar β y de otro árbol γ, que puede ser un árbol inicial, auxiliar o derivado de adjunciones realizadas previamente. En su forma más simple, una adjunción puede tener lugar si la etiqueta de un nodo del árbol (denominado nodo de adjunción) coincide con la etiqueta del nodo raíz de un árbol auxiliar . En tal caso, el árbol derivado resultante se construye como sigue:

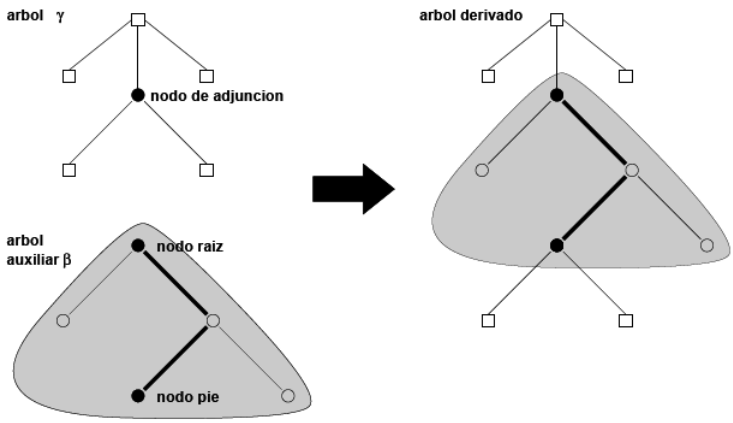

Figura **2-3**: Operación de adjunción [37, 38]

1. El subárbol de γ dominado por el nodo de adjunción se separa de γ, dejando una copia del nodo en γ.

2. El árbol auxiliar β se pega a la copia del nodo de adjunción de manera que la raíz del árbol auxiliar se identifica con dicha copia.

3. El subárbol separado de γ se une al nodo pie del árbol auxiliar de tal modo que la raíz del subárbol separado se identifica con el nodo pie de β.

La aplicación de la operación de adjunción depende de las etiquetas de los nodos. Sin embargo, se puede especificar para cada nodo un conjunto de restricciones que permite indicar con mas precisión los árboles auxiliares que pueden ser adjuntados. Las restricciones asociadas a un nodo, que se denominan *restricciones de adjunción*, son clasificados como sigue:

- Restricciones de *adjunción selectiva* (SA): especifican el subconjunto de árboles auxiliares que pueden participar en una operación de adjunción.

- Restricciones de *adjunción nula* (NA): impiden la realización de adjunciones

- Restricciones de *adjunción obligatoria* (OA): especifica un subconjunto de árboles auxiliares, uno de los cuales ha de ser utilizado obligatoriamente en una operación de adjunción.

La gramática de adjunción de árboles definen el lenguaje conformado por el conjunto de cadenas $w \in V_T^*$ tal que w constituye la frontera de un árbol derivado a partir de un árbol inicial.

2.1.3. Árboles de derivación

Los árboles derivados de las gramáticas libres de contexto contienen información sobre operaciones realizadas sobre los nodos. Pero en las gramáticas libres de contexto de acuerdo con [17, 21] los árboles derivados muestran la manera como ha sido construido un árbol indicando las operaciones de adjunción que han sido realizadas junto con el árbol auxiliar involucrado. La raíz de un árbol de derivación debe estar etiquetada por un árbol inicial. Los demás nodos del árbol de derivación estan etiquetados por un árbol auxiliar y por el nodo en el que se realizó la adjunción. Por lo general se utilizan direcciones de Gorn [15] para referirse a los nodos de un árbol elemental. En el direccionamiento de Gorn se utiliza 0 para referirse al nodo raíz, k para referirse al k-ésimo hijo del nodo raíz y p. q para referirse al q-ésimo hijo del nodo con dirección p. Las gramáticas de adjunción de árboles son interesantes desde el punto de vista lingüístico por la extensión del dominio de localidad, la factorización de la recursión en el dominio de dependencias, la posibilida de representar dependencias cruzadas y su carácter lexicalizado.

2.2. Modelos probabilísticos

Los modelos probabilísticos son modelos matemáticos que describen el comportamiento de los datos que pueden ser observados a partir de un sistema. Collins [12] define un modelo probabilístico para la predicción de árboles sintácticos basados en el entrenamiento y manejo de los datos observados de otros árboles almacenados en el corpus.

Si un conjunto ς es un espacio de eventos discretos, y P es una distribución de probabilidad sobre este espacio, entonces

- $0 \leq P(A) \leq 1$ para todo $A \in \varsigma$
- $\sum_{A \in \varsigma} P(A) = 1$

P es una función parametrizada, y se escribirá $P(A|\theta)$, para definir la probabilidad de un evento A dado un vector de parámetros θ

- El espacio de parámetros Ω es un espacio de parámetros.
- $P(A|\theta)$ es una medidad de probabilidad ς

Como por ejemplo se toma el caso de lanzar una moneda que tiene dos posibilidades C (Cara) ó S (Sello). Y donde la probabilidad de que salga cara es p. En este caso:

- El espacio de eventos ς es el conjunto $\{C, S\}$.
- El vector de parámetros θ tiene un único componente, p.

- La medida de probabilidad $P(A|\theta)$ es definida como p si $A = C$, $1 - p$ si $A = S$.

- El espacio de parámetros es el conjunto Ω es el conjunto $[0, 1]$. (p debe tomar un valor entre 0 y 1 para $P(A|\theta)$ ser una medida de probabilidad).

Estimación de máxima similitud

Según Collins [11] se puede definir una secuencia de n eventos como $X =< X_1, X_2 \ldots, X_n >$. Por ejemplo, podemos tirar la moneda 5 veces y obtener $X =< C, S, S, C, S >$. Un método muy general es usar la estimación de máxima similitud. Asumiendo que los eventos son independientes entre ellos, la función de similitud, L es definida asi:

$$L(X|\theta) = \prod_{i=1,\ldots n} P(X_i|\theta)$$

La estimación de máxima similitud θ'_{ML} es el valor de θ (en el espacio de parámetros Ω) que maximiza la función de similitud:

$$\theta'_{ML} = arg\, max_{\theta \in \Omega} L(X|\theta)$$

En esta parte se definen los modelos probabilísticos sobre las gramáticas libres de contexto iniciando con las PCFG (Gramáticas probabilisticas libres de contexto) que son CFGs con una distribución de probabilidad que sirve para el tratamiento de la ambigüedad de sentencias. De otra parte, también se definen las PCFG lexicalizadas que adicionan estructuras contextuales lexicalizadas y dependencia de constituyentes para el tratamiento contextual sintáctico de las sentencias. Cabe anotar que la estructura principal para este trabajo basado en modelos probabilísticos son las CFGs.

Gramáticas Probabilísticas Libres de Contexto (PCFG)

Un aprendizaje probabilístico para el análisis sintáctico puede ser definido como sigue:

- El espacio de entrada X es un conjunto de secuencias $< w_1, \ldots, w_n >$donde cada w_i es obtenido de un conjunto de "palabras" V.

- El espacio de salida Y es un conjunto de árboles de análisis generados por el misma gramática libre de contexto. Cada árbol de análisis tiene un miembro de V^*

- En orden de definir el función de mpaeo $f : X \rightarrow Y$ se define un score de distribución $X \times Y \rightarrow [0, 1]$. Las PCFGs pueden ser usadas para definir una probabilidad $P(x, y—\theta)$ sobre el espacio de posibles pares de sentencias/árboles de análisis.

Definición Formal

Una gramática libre de contexto probabilística es usualmente definida como una 4-tupla $G = (N, \Sigma, P, S)$ donde:

- N es un conjunto finitio de símbolos no terminales.
- Σ es un conjunto finito de símbolos terminales.
- P es un conjunto finito de reglas de producción y escritura. Las reglas en P toman la forma $\alpha \to \beta$ como $P(\alpha \to \beta|\alpha)$. Esto es interpretado como una probabilidad condicional de escoger la regla $\alpha \to \beta$, dado que α es el no terminal siendo reescrito en una derivación. Si D es una función de asignando una probabilidad para cada miembro de P, una PCFG es una 5-tupla $G = (N, \Sigma, P, S, D)$.

Dado una PCFG, la probabilidad para cualquier árbol libre de contexto en el lenguaje es el producto de probabilidades para las reglas que este contiene. Si T es una derivación libre de contexto que involucra n reglas de la forma $\alpha_i \to \beta_i$.

$$P(T) = \prod P(\alpha_i \to \beta_i|\alpha_i)$$

Una PCFG define una distribución de probabilidad sobre cadenas de caracteres. If $\mathcal{T}(S)$ es el conjunto de árboles donde el exterior de la cadena es S, entonces:

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T)$$

PCFG Lexicalizadas

En lugar de modificar la gramática, se modifica el modelo probabilístico del PCFG, introduciendo información léxica. En este caso el núcleo del constituyente (head) que es la palabra que determina las propiedades sintácticas del constituyente. Por ejemplo para un grupo verbal (grup.verb) el núcleo es el verbo. Se construye entonces un modelo probabilístico tomando en consideración los núcleos léxicos de los constituyentes. Una PCFG puede ser lexicalizada por la asociación de una palabra w y un etiqueta t (part-of-speech POS) con cada no terminal X en el árbol sintáctico. El modelo PCFG puede ser aplicado a estas reglas lexicalizadas de la misma manera como con las reglas de PCFG.

Modelos basados en historias

En Collins [12, 11] se define un mapeo de cada miembro $\mathcal{X} \times \mathcal{Y}$ a una secuencia de decisiones $(d_1, d_2, \ldots, d_n)$. La probabilidad conjunto de un miembro (x, y) de $\mathcal{X} \times \mathcal{Y}$ es escrita usando una regla de cadena de probabilidades:

$$P(x, y) = P(\langle d_1, d_2, \ldots, d_n \rangle) = \prod_{i=1\ldots n} P(d_i|d_1 \ldots d_{i-1})$$

El contexto condicional para cada $d_i, \langle d_1, d_2, \ldots, d_{i-1} \rangle$, es referida como la "historia", y es equivalente a alguna estructura construida parcialmente.

El mapeo entre eventos en $X \times Y$ y una secuencia de decisiones es registrada definiendo un algoritmo estocástico que genera eventos en $X \times Y$. Este algoritmo usa ciertos puntos que hacen una escogencia aleatoria entre decisiones presentadas como alternativas, de acuerdo a una distribución de probabilidad. La traza del algoritmo puede ser representada como una secuencia que se está construyendo. la probabilidad de esta secuencia de decisión es el producto de probabilidades de diferentes decisiones.

2.2.1. Modelo de Collins

A continuación se presenta los modelos propuestos por Collins en [11].

Modelo Básico

Lo primero que se nota en cada regla de una PCFG Lexicalizada es la forma:

$$P(h) \rightarrow L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m)$$

Donde H es el núcleo sintáctico de la frase, el cuál hereda la palabra-núcleo h, de su nodo padre P. $L_1 \ldots L_n$ y $R_1 \ldots R_m$ son modificadores de izquierda y derecha del núcleo H. Tanto n como m pueden ser cero, Si $n = m = 0$, son reglas unarias.
La generación de RHS de cada regla, dada el LHS, ha sido descompuesta en tres pasos:

1. Generación de la etiqueta del núcleo constituyente de la frase, con probabilidad $P_H(H|P,h)$.
2. Generación de modificadores a la izquierda del núcleo con probabilidad $\prod_{i=1\ldots n+1} P_L(L_i(l_i)|P,h,H)$, donde $L_{n+1}(l_{n+1}) = STOP$. El símbolo $STOP$ es adicionado al vocabulario de no-terminales, y al modelo para ir generando modificadores a la izquierda donde ha sido generado el constituyente con el núcleo.
3. Generación de modificadores a la derecha del núcleo con probabilidad $\prod_{i=1\ldots n+1} P_R(R_i(r_i)|P,h,H)$, donde $R_{m+1}(r_{m+1})$ es definida como $STOP$.

Modelo 1

Se adicióna el concepto de distancia al modelo, extendiendo el modelo basado en historias, una extensión que usa el modelo de distancias. En general, a través de cada modificador se puede depender de una función ϕ de previos modificadores, categorías núcleo/padre y palabras-núcleo.

$$P_l(L_i(l_i)|L_1(l_1)\ldots L_{i-1}(l_{i-1}), P(h), H) = P_l(L_i(l_i)|\phi(L_1(l_1)\ldots L_{i-1}(l_{i-1}), P(h), H))$$

$$P_r(R_j(r_j)|L_1(l_1)\dots L_{n+1}(l_{n+1}), R_1(r_1)\dots R_{j-1}(r_{j-1}), P(h), H) =$$
$$P_r(R_j(r_j)|\phi(L_1(l_1)\dots L_{n+1}(l_{n+1}), R_1(r_1)\dots R_{j-1}(r_{j-1}), P(h), H))$$

Asumiendo que los modificadores son generados independientemente, la distancia puede ser incorporada en el modelo incrementando la cantidad de dependencias entre estos modificadores. Si el orden de derivación es corregido para ser primero en profundidad, el modelo puede condicionar en cualquier estructura abajo de modificadores:

$$P_l(L_i(l_i)|H, P, h, L_1(l_1)\dots L_{i-1}(l_{i-1})) = P_l(L_i(l_i)|H, P, h, distance_l(i-1))$$

$$P_r(R_i(r_i)|H, P, h, R_1(r_1)\dots R_{i-1}(r_{i-1})) = P_l(R_i(r_i)|H, P, h, distance_r(i-1))$$

donde $distance_l$ y $distance_r$ son funciones de la cadena de superficie abajo de modificadores previos. La medida de distancia es un vector con los 2 siguientes elementos:

1. La cadena es de longitud cero.
2. La cadena contiene un verbo.

Modelo 2

El segundo modelo [11] corresponde a una distinción entre adjunción/complemento y subcategorización. Cuando se identifican los complementos se marcan con el sufijo "-C" en los no-terminales. En la figura **2-4** se muestra un ejemplo teniendo a "IBM" como complemento y a "Last week" como un adjunto. La etapa de preprocesamiento puede adicionar este detalle a la salida del analizador, pero aún así hay dos buenas razones para hacer la distinción:

1. Identificar los complementos es suficientemente complejo para garantizar un tratamiento probabilístico. La información léxica es necesaria.
2. Hacer la distinción entre adjunción/complemento mientras se hace el análisis, puede ayudar a mejorar la precisión.

El modelo puede aprender las propiedades lexicas disguiendolas entre complementos y adjuntos. Por lo tanto este puede sufrir de malos supuestos. El proceso generativo es extendido para incluir una escogencia probabilística de marcos de subcategorización del lado izquierdo, y del lado derecho del núcleo.

- Escoger el núcleo (cabeza) H con probabilidad $P_H(H|P, h)$.
- Escoger los marcos de subcategorización del lado izquierdo y el lado derecho del núcleo, LC y RC con probabilidades $P_{LC}(LC|P, H, h)$ y $P_{RC}(RC|P, H, h)$, Cada marco de subcategorización es un multiconjunto especificando los complementos que son requeridos por el núcleo en su lado izquierdo y derecho.

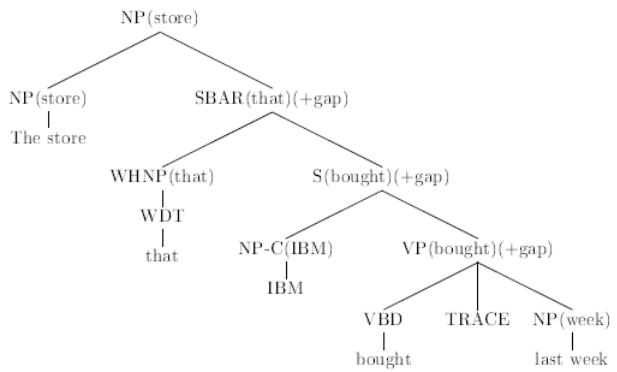

Figura **2-5**: Ejemplo de un árbol mostrando la brecha (gap) [12]

- Generar los modificadores del lado izquierdo y derecho con probabilidades

$$P_l(L_i, l_i|H, P, h, distancia(i-1), LC)$$
y
$$P_r(R_i, r_i|H, P, h, distancia(i-1), RC)$$

respectivamente. Estos requerimientos de marcos de subcategorización son adicionados al contexto condicionado. Cuando los complementos son generados, estos son removidos del multiconjunto de subcategorizaciones.

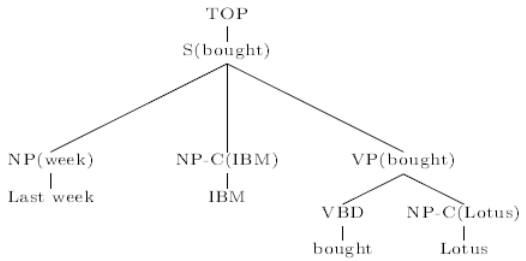

Figura **2-4**: Ejemplo de un árbol mostrando el complemento [12]

Modelo 3

Otro obstáculo para extraer la estructura de argumentos-predicados de los árboles de análisis sintáctico es el movimiento del núcleo sintáctico. Las frases de nombre son extraidas de la posición de sujeto y la posición del objeto. Esto puede hacerse escribiendo patrones

basados en reglas que identifican trazas en un árbol sintáctico. Esta tarea es también lo suficientemente compleja para garantizar un tratamiento probabilístico y la integración puede ayudar a mejorar la precisión del análisis sintáctico. Otra razón para un tratamiendo integrado de trazas es mejorar la parametrización del modelo. En particular, las probabilidades de subcategorización son marcadas por extracción. El análisis de Collins es generativo, de estructura de frase y dirigido por el núcleo sintáctico, lo que corresponde a una gramática de este tipo llamadas gramáticas HPSG que son las siglas de *Head-driven Phrase Structure Grammar*. Esta grámatica es una extensión de las gramáticas de estructura de frase generativas (GPSG). Existen formalismos similares a GPSG que manejan movimientos de núcleos sintácticos adicionando una característica de diferencia (brecha) para cada no-terminal en el árbol y propagando estas diferencias a través de los árboles hasta que finalmente queda marcado una traza de complemento, tal como lo muestra la figura **2-5**. Dado que las LHS de la regla tiene una brecha (gap), hay tres formas de que esta brecha sea pasada hacia abajo a el RHS. **Head** el gap es pasado al núcleo de la frase, como una regla. **Left, Right** El gap es pasado recursivamente a uno de los modificadores de la izquierda o derecha del núcleo, o marcado como un argumento de traza a la izquierda o derecha del núcleo. Se especifica un parámetro $P_G(G|P,h,H)$ donde G es otro **Head**, **Left** o **Right**. El proceso generativo es extendido a escoger entre los casos despues de generar el núcleo sintáctico. El resto de la frase es generada en diferentes formas dependiendo de como el gap es propagado. En el caso del **Head** los modificadores de iquierda y derecha son generados como normal. En el caso de **Left, Right** un gap requerido es adicionado a otro de variable SUBCAT izquierda o derecha.

2.2.2. Implementación de Bikel

Dan Bikel en su tesis doctoral [4] desarrolló técnicas y metodologías para examinar sistemas complejos como los modelos de análisis sintáctico probabilísticos lexicalizados. La primera idea es tratar el modelo como datos, el cual no es un método particular, pero es un paradigma y metodología de investigación. Para conseguirlo construye un motor de análisis sintáctico multi-lenguaje con la capacidad de instanciar una gran variedad de modelos analizadores probabilísticos. Como modelo línea base apropiado escoge instanciar los parámetros del segundo modelo de Collins. Bikel identifica once pasos de preprocesamiento necesarios para preparar los árboles de entrenamiento cuando se usa el modelo de análisis de Collins.

1. Eliminación nodos innecesarios
2. Adición nodos base NP
3. Reparación NPs base
4. Adición de información (solo aplicable al modelo 3)

5. Re-etiquetación de sentencias

6. Removiendo elementos nulos.

7. Levantando puntuación.

8. Identificación de argumentos no-terminales

9. Eliminación de terminales no usados.

10. "Reparación" Oraciones sin sujeto.

11. Encontrar núcleos sintácticos.

El orden presentado no es arbitrario, cada uno de esos pasos depende del resultado obtenido en el paso anterior. Se separan los pasos en unidades funcionales: Una implementación puede combinar pasos que son independientes uno del otro. Finalmente, notamos que el paso final, encontrar núcleos sintácticos, es actualmente necesario por alguno de los pasos previos en ciertos casos; en la implementación, se emplea selectivamente un módulo de encontrar los núcleos durante los primeros 10 pasos cuando es necesario.

Reglas para encontrar el núcleo sintáctico

Collins define en su trabajo [12] reglas heurísticas para encontrar el núcleo sintáctico de una frase del idioma inglés basado en las reglas de cabeza de Magerman [22]. Estas reglas no son aplicables al idioma español por las diferencias gramaticales que existen entre los dos idiomas, y se hace necesario crear reglas para buscar los núcleos de los componentes sintácticos o constituyentes según las reglas que usa el corpus Ancora. Basados en la reglas de Magerman Brooke Cowan en su trabajo de doctorado [14] define un conjunto de reglas para encontrar el núcleo sintáctico que son usadas para modelos de análisis sintáctico para el español como también lo hace Chrupala [10]. Este conjunto de reglas determinísticas que se muestran en la figura **2-6** especifican cual de los hijos de un nodo padre es el núcleo. Las demás reglas se definen basadas en la gramática del idioma español.

Non-Terminal	Dir	Headchild
CONJ	left	CS
COORD	left	CC
GRUP	right	N,W,P,Z,AQ,GRUP
INC	left	S
INFINITIU	left	V
INTERJECCIO	left	I
NEG	left	RN
PREP	left	SP
SA	left	AO,AQ
SN	right	GRUP
	left	SN
	left	S,RELATIU,SP
	right	N
SP	left	PREP
	left	SP
SADV	left	RG
	left	P
	left	SADV
s	right	AO,AQ
	left	s
S	left	GV
	left	INFINITIU
	left	S
	left	GERUNDI
SBAR	left	CP
*CC1	left	CC2
*CC2	right	any child
*CP	left	RELATIU-CP,CONJ-CP
	right	any child

Figura **2-6**: Reglas de cabeza para el español [14]

2.3. Tecnologías de NLP usadas

2.3.1. Ancora Corpus

ANCORA [24] (ANnotated CORpora) es un corpus del catalán (AnCOra-CA) y español (AnCOra-ES) con diferentes niveles de anotación. Cada corpus contiene 500.000 palabras que han sido construidas de manera incremental a través de trabajos previos como el corpus 3LB: 3LB-CAT y 3LB-ESP, cada uno con 100.000 palabras correspondientes a 4.000 oraciones para el español y 2.600 oraciones para el catalán. Ambos corpus estan automáticamente etiquetados con información morfosintáctica y chequeada manualmente. Estos han sido ampliamente usados como corpus de entrenamiento para sistemas de aprendizaje, sistemas basados en reglas y sitemas de etiquetamiento (Pos Tagging). Los corpus 3LB son sintácticamente etiquetados con constituyentes y funciones de una manera manual. La infor-

```
(
 (S (r ao) (r aq) (l s) (r grup.verb) (r sn) (l gv ) (l infintiu ) (l gerundi) )
 (grup.verb (r infinitiu) (r gerundi) (r vmp) (r vsp) (r vap) (r vmi))
 (grup.nom (l n)  (l grup.nom)  (l p)  (l w)  (l z)  (l a)  (l v)
      (l s.a) (l participi) (l participle) (l relative) (l sp) (l sn)
 )
 (sn (r grup.nom) (r grup) (l sn) (l s) (l relatiu) (l sp) (r n))
 (sp (l prep) (l sp))
 (sadv (l grup.adv) (l rg) (l p) (l sadv))
 (espec.fp (l da0mp0) (l da0fs0) (l di0ms0) (l da0fp0))
 (snd (l grup.nom))
 (sa (l grup.a))
 (sno (l grup.nom))
 (conj.subord (l cs))
 (prep (l sps00))
 (espec.fs (l da0fs0))
 (sn.x (l grup.nom))
 (espec.ms (l da0ms0))
 (espec.mp (l Z))
 (conj (l cs))
 (coord (l cc))
 (grup (r N) (r W) (r P) (r Z) (r aq) (r grup))
 (inc (l s))
 (infinitiu (l v))
 (interjeccio (l i))
 (neg (l rn))
 (prep (l sp))
 (sa (l ao) (l aq))
 (sbar (l cp))
 (* (r))
)
```

Figura **2-7**: Reglas para encontrar el núcleo sintáctico usadas en la implementación de Bikel para el español.

mación lingüística anotada en ANCORA se encuentra en formato XML y en formato TBF (Treebank Bracketted Format). Este último formato es propio para analizadores sintácticos, siendo XML un formato más abierto para otro tipo de información lingüística. AnCOra es el resultado de extender los corpus 3LB-CAT/ESP sobre 500.000 palabras y enriquecida con información semántica en diferentes niveles: estructuras de argumentos, roles temáticos, clases semánticas, entidades nombradas (NE) y sentidos nominales. Además de la anotación sintáctica y anotación manual semántica, el corpus de Ancora fué anotado automáticamente en su parte morfológica, que consitió en la asociación de un lema, categoría y atributos morfológicos a cada palabra del corpus. Este anotación morfológico implica analizar cada pieza del corpus y asignarle todas las posibles etiquetas que pueda recibir. En el proceso de desambigüación se selecciona solo una etiqueta. El conjunto de etiquetas usado por Ancora (corpus con árboles sintácticos) y Freeling (Herramienta de etiquetamiento de las palabras) que se explica en la siguiente sección, es el propuesto por el grupo EAGLES [19] para la anotación morfosináctica de lexicones y corpus para todas las lenguas europeas.

Palabra	Lema 1	Etiqueta 1	Lema 2	Etiqueta 2	Lema 3	Etiqueta 3	Lema 4	Etiqueta 4
bajo	bajar	VM1SIP	bajo	AQ0MS	bajo	P000	bajo	NCMS, ...

Tabla **2-1**: Ejemplos de etiquetas usadas en el corpus de Ancora [24]

La palabra bajo puede ser la tercera forma singular del verbo bajar (VM1SIP, que de acuerdo al conjunto de etiquetas de EAGLE significa que es un verbo principal en primera persona en tiempo pasado), o el adjetivo corto, pequeño, chico (AQ0MS según EAGLE es un adjetivo calificativo masculino y singular), o la preposición bajo, sobre (P000) o un instrumento musical (NCMS, Nombre común masculino y singular). En el corpus se realiza una distinción entre complementos y adjuntos, por lo tanto los nodos que contienen un sujeto, un verbo, los complementos del verbo y los adjuntos son nodos hermanos. Es comúnmente aceptado que dos oraciones tengan dos constituyentes principales: el sujeto y el predicado, luego incluyendo el verbo, sus argumentos y sus adjuntos. La relación entre el verbo y sus argumentos es más cerrada que la relación entre el verbo y sus adjuntos. La anotación semántica especifica si un constituyente es argumento o no. En la figura **2-8** se muestra una anotación sintáctica de una sentencia S *"La declaración propugnó trabajar por la igualdad social."* En este trabajo además del uso de la información morfológica, léxica y sintáctica, también se utilizan los diferentes tipos de información semántica: a) las clases semánticas y estructuras de argumentos de predicados verbales, donde la relación entre predicados y argumentos se expresa mediante roles temáticos. b) Entidades nombradas, tanto fuertes como débiles.

```
(S
    (sn-SUJ
      (espec.fs
        (da0fs0 La el))
      (grup.nom.fs
        (ncfs000 declaración declaración)))
    (grup.verb
      (vmis3s0 propugnó propugnar))
    (S.NF.C.co-CD
      (S.NF.C
        (infinitiu
          (vmn0000 trabajar trabajar))
        (sp-CC
          (prep
            (sps00 por por))
          (sn
            (espec.fs
              (da0fs0 la el))
            (grup.nom.fs
              (ncfs000 igualdad igualdad)
              (s.a.fs
                (grup.a.fs
                  (aq0cs0 social social))))))))
(Fp . .))
```

Figura **2-8**: Árbol sintáctico completo anotado con constituyentes [38]

2.3.2. Freeling

Freeling [6] es una biblioteca de procesamiento de lenguajes de código abierto que provee una gran variedad de analizadores para varios idiomas. Es desarrollada y mantenida por el centro de investigación TALP (Center for Language and Speech Technologies and Applications - http://www.talp.upc.edu/) . El sistema completo esta escrito en C++, distribuido sobre una licencia LGPL, que facilita su portabilidad a nuevos idiomas y la personalización a las necesidades de uso.

La arquitectura de Freeling 3.0 [28] esta organizada en módulos de máquinas de aprendizaje, algoritmo de extracción de características y algoritmos de clasificación. En la versión 3.0 han extendido el repertorio de algoritmos con máquinas de soporte vectorial, usando el proyecto de código abierto libSVM. El código de libSVM ha sido integrado a Freeeling sobre una capa común a los demás clasificadores.

Freeling realiza el análisis sintáctico de constituyentes usando chart parser y el análisis de dependencias usando el analizador de dependencias de Txala [28]. En este trabajo se usa Freeling para etiquetar cada palabra usando POS tagging para la frase de entrada. Esta frase etiquetada es la entrada para el analizador sintáctico.

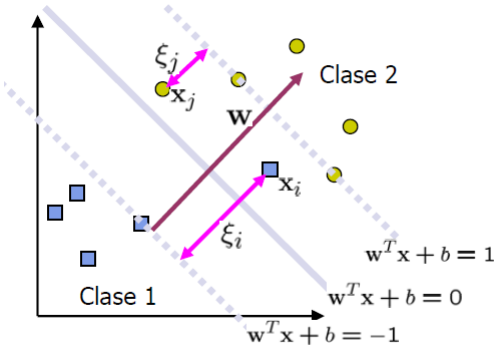

Figura **2-9**: Hiperplano mostrando los vectores de soporte [13]

2.4. Máquinas de Vectores de Soporte

Las máquinas con vectores de soporte son un sistema de aprendizaje basado en la teoría del aprendizaje estadístico desarrollado inicialmente por V. Vapnik [36]. Algunas de las aplicaciones de clasificación o reconocimiento de patrones son el reconocimiento de firmas, reconocimiento de imágenes como rostros, categorización de textos y en este trabajo se usa para hacer clasificación de argumentos de verbo. Su técnica se basa en un kernel el cual es una función de transformación de espacios vectoriales que mapea los datos a un espacio de características de mayor dimensión buscando la máxima separación entre clases, propiedad conocida como margen amplio, y que aumenta la capacidad computacional de la máquina de aprendizaje lineal.

Las máquinas de vectores de soporte tienen la ventaja que pueden ser usadas para resolver problemas lineales y no lineales. La importancia de estas máquinas radica en la precisión de predicción sobre pocos ejemplos y una gran cantidad de características (features).

Margen amplio

Las SVMs son clasificadores de margen amplio que permiten encontrar el hiperplano lineal más probable ajustado a un conjunto de ejemplos que han sido linealmente separables mediante el uso de kernels. Una representación gráfica del proceso de marginalización sobre un conjunto de vectores es mostrado en la figura **2-9**. Para m observaciones hay un par de datos anotados.

- Un vector $x_i \in R^n, i \ldots, m$
- Una etiqueta $y_i \in \{+1, -1\}$

Suponiendo que se tiene un hiperplano que separa las muestras positivas $(+1)$ de las negativas (-1). Los puntos x_i que estan en el hiperplano satisfacen $w.x + b = 0$.

W es normal al hiperplano. $\frac{|b|}{\|w\|}$ es la distancia perpendicular del hiperplano al origen. $\|w\|$es la norma euclídea de w. Lo que se busca es separar los puntos de acuerdo al valor de su etiqueta y_i en dos hiperplanos diferentes como se muestra en la figura **2-9** maximizando el margen creado entre estos hiperplanos.

El Kernel

Una forma de clasificar de manera simple los datos es a través de una línea recta, un plano recto o un hiperplano N-dimensional. Por lo general los universos a estudiar no suelen presentar casos de dos dimensiones como en el ejemplo mostrado en la figura **2-9**, en su lugar el algoritmo SVM debe tratar con mas de dos variables predictoras, casos donde los conjuntos de datos ni pueden ser completamente separados o clasificaciones con más de dos categorías, y debido a las limitaciones computacionales de las máquinas de aprendizaje lineal, éstas no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real. Para ofrecer una solución a este problema se usan funciones Kernel que proyectan la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de la máquinas de aprendizaje lineal. Es decir, se hace un mapeo del espacio de entrada X a un nuevo espacio de características de mayor dimensionalidad:

$$F = \{\phi(x)|x \in X\}\ x = \{x_1, x_2, \ldots, x_n\} \rightarrow \phi(x) = \{\phi(x)_1, \phi(x)_2, \ldots, \phi(x)_n\}$$

A continuación se presentan los tipos de funciones Kernel usados para la este mapeo.

- Lineal: $K(x_i, x_j) = X_i^T X_j$
- Polinomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0.$
- Función de Base Radial (RBF): $K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}, \gamma > 0$
- Sigmoidal: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$

Donde γ, r y d son parámetros de la función kernel.

2.5. Clasificación de argumentos de verbo

Una oración en inglés o español puede descomponerse en partes llamados sintagmas. Todo sintagma posee un único núcleo sintáctico, siguiendo la naturaleza que nos presenta una organización jerarquizada como lo hace la lingüística. El núcleo es un morfema o palabra que determina las propiedades sintácticas y combinatorias del sintagma al que pertenece. El núcleo es el constituyente hijo que posee la misma distribución que el constituyente madre. El núcleo no implica relación de subordinación o dependencia. Se ha requerido el desarrollo

de herramientas que permitan búsquedas más precisas a preguntas que giran alrededor del núcleo, en este caso el verbo. Preguntas como ¿qué ocurrió?, ¿en donde?, ¿quién?, ¿cómo? y ¿cuando?. En otras palabras la identificación de los argumentos del verbo.

Un role semántico es la relación entre un constituyente sintáctico (generalmente, aunque no siempre argumento de verbo) y un predicado (generalmente aunque no siempre, un verbo) [30]. Según la teoría de Tesniere [35] se considera que los elementos fundamentales de la oración son: los actores, el verbo que representa la acción, y los complementos del verbo. Los actores son el primer argumento del verbo, encargado de realizar la acción. Como complementos tenemos el objeto sobre el cual se realiza la acción, quien recibe los beneficios de la acción que es identificado como un complemento indirecto de la oración y por ende es otro argumento del verbo, y un complemento circunstancial que es considerado como un elemento opcional que amplia el significado de la oración. Existen trabajos que realizan esta clasificación e identificación de argumentos usando máquinas de vectores de soporte con características lexicas y sintácticas. Gildea y Jurafsky realizan un etiquetamiento automático de roles semánticos Pradham [30, 29] propone un algoritmo de máquinas de aprendizaje basado en máquinas de vectores de soporte para hacer análisis semántico superficial extendiendo el trabajo de Gildea y Jurafsky. Reyes [31, 32] también hace clasificación de roles semánticos definiendo características sintácticas, semánticas y contextuales. Por otro lado Chrupala [10] también usa máquinas de vectores de soporte para la asignación de etiquetas de función a sentencias analizadas con el algoritmo de Bikel entrenadas igualmente con un subconjunto del corpus ANCORA, Cast3LB. A continuación se describen las características usadas por los diferentes autores (Pardhan, Reyes y Chrupala) para los diferentes clasificadores semánticos relacionados con argumentos de verbo. En esta parte se definen las características usadas por Pradham para su clasificador semántico superficial [30, 29], estas características se basan por otras definidas por Jurafsky [18].

- **Predicado**: El lema del predicado es usado como característica.
- **Path**: La ruta sintáctica a través del árbol de análisis del constituyente al predicado clasificado.
- **Tipo de frase**: Esta es la categoría sintáctica.
- **Posición**: Característica binaria identificando si la frase esta antes o despues del predicado.
- **Palabra-cabeza**: La cabeza sintáctica de la frase.
- **Subcategorización**: Esta es la regla de estructura de frase expandiendo el nodo padre del predicado en el árbol de análisis.

2.5.1. Clasificación de Roles semánticos

Reyes [31] mediante la tarea de clasificación identifica el tipo de role semántico existente entre un evento y sus actantes a las que define como pares evento-entidad y la tarea consiste en determinar que tipo de rol semántico expresa la entidad. Las siguientes son las características sintácticas, semánticas y contextuales definidas para el clasificador. Las características sintácticas describen estructura, posición, información morfológica con la ayuda de la herramienta Freeling.

- **Posición de la entidad**: la entidad puede estar a la izquierda o derecha del núcleo.
- **Distancia de la entidad**: el número de palabras existentes entre la entidad y el núcleo.
- **Información morfológica de la entidad**: modo, tiempo, persona y número.
- **Longitud de la entidad**: la secuencia de los n elementos de los n-gramas.
- **Información morfológica**: Del núcleo de la entidad.
- **Entidad definida**: una entidad se considera definida si su artículo es definido.
- **Longitud del núcleo**: secuencia de los n-elementos de los n-gramas del núcleo.

Las características semánticas describen el significado y sentido de los pares evento:entidad.

- **Entidad nombrada**: Las entidades nombradas son detectadas con el módulo de Reconocimiento de Entidades Nombradas de la herramienta FreeLing.
- **Tipo de preposición**: Si la entidad pertenece a una frase preposicional, se determina el tipo de preposición.
- **Tipo de entidad**: Las entidades se categorizan con respecto a su núcleo nominal, nombre propio o nombre común.
- **Hiperónimo de la entidad**: La relación de hiperonimia ayuda a determinar el tipo de entidad en función de su rasgo semántico, se obtienen los hiperónimos de la entidad en tres niveles y se verifica si es un tipo de agente animado u objeto.

Las características contextuales describen los pares evento-entidad considerando las palabras que ocurren en el contexto de la entidad, con un tamaño de ventana determinado ($n = 3$).

- **Contexto izquierdo**: Las tres palabras a la izquierda representan su contexto izquierdo y se extrae su categoría gramatical (adjetivo, adverbio, determinante, sustantivo, verbo, pronombre, conjunción, preposición o signo de puntuación) para cada una.

- **Contexto derecho**: Las tres palabras a la derecha representan su contexto derecho y se extrae su categoría gramatical (adjetivo, adverbio, determinante, sustantivo, verbo, pronombre, conjunción, preposición o signo de puntuación) para cada una.

Otro de los trabajos importantes en la clasificación de etiquetas funcionales sintácticas usando información semántica es el de Chrupala [10] en el cual se define los siguientes tres tipos de características codificando información léxica, morfológica y de estructura para el nodo a ser etiquetado y los nodos del contexto vecino.

- **Características de Nodo**: Posición relativa a la núcleo, lema del núcleo, categoría.
- **Características locales**: persona del verbo, número del verbo, categpría del nodo padre del verbo.
- **Características contextuales**: Son las características de nodo de los dos nodos hermanos anteriores y siguientes, si estos existen.

3 Clasificador de argumentos de verbos

En esta sección se define el clasificador de argumentos de verbo usando Máquinas de vectores de soporte (SVMs). Los argumentos de verbo clasificados servirán de insumo para la anotación sintáctica de las sentencias de entrada. En esta parte son también descritas las características y la forma de selección basados en conceptos semánticos. El uso de clasificadores y estructuras lingüísticas complejas en la construcción de los analizadores sintácticos probabilísticos es una posibilidad intermedia para el mejoramiento y comprensión de los PCFGs lexicalizados. Trabajos como el de Pradhan & Jurafsky [30] utilizan SVMs para clasificación automática de argumentos para hacer análisis semántico superficial. En esta tesis se pretende el uso de los clasificadores de argumentos de verbos para la anotación sintáctica de una sentencia dada. También se pretende anotar la información lingüística en el árbol sintáctico de salida el cual se propaga a través del uso del clasificador de argumentos de verbo. En esta parte se definen las características que que son usadas para la implementación del clasificador de argumentos.

3.0.2. Características de los argumentos de verbo

Con base en las características definidas en los trabajos presentados por Jurafsky [18], Pradham [30], Reyes [31] y Chrupala [10], se definen a continuación las características para la implementación del clasificador de argumentos de verbo que es usado en el analizador sintáctico probabilístico de Bikel [4], como un refuerzo del segundo modelo de Collins [11]. Estas características fueron escogidas primero por su facilidad de implementación y segundo porque se consideran relevantes para determinar si un ítem constituyente es un argumento verbal o no. Las características son obtenidas a partir de los subárboles que representan constituyentes y conforman el árbol sintáctico de la oración. Son obtenidas en el momento de unir el subárbol, con otro que hace referencia a un grupo verbal. Esto para poder identificar el nodo constituyente como argumento de verbo.

- **Posición del núcleo**: Obtener la posición del núcleo sintáctico. Cada subárbol representa un sintagma o constituyente de la oración, para obtener esta característica se implementó una función que determina de izquiera a derecha empezando en 1, la posición del núcleo del sintagma representado.
- **Codificación de etiqueta** Asignación de un código para la etiqueta del nodo raíz del árbol. Las etiquetas del nodo raíz son codificadas y convertidas a su representación

binaria, como una distinción característica del subárbol.

- **Aridad**: Número de hijos del nodo padre. Se refiere a la anchura del segundo nivel del árbol.

- **Anchura**: Número de nodos hojas. Se refiere al número de palabras que conforman el constituyente sintáctico.

- **Longitud**: Longitud del árbol. Corresponde al máximo número de nodos que existen entre el nodo raíz y una hoja del árbol. Se calcula la cantidad de nodos desde la raíz a las hojas del cual es seleccionado el mayor valor.

- **Distancia**: Número de palabras entre el nodo del constituyente modificador y el verbo.

3.1. Ajuste de parámetros

Para este trabajo se decidió usar la librería *libsvm* provista en [7] para la implementación del algoritmo de clasificación con máquinas de vectores de soporte. Esta implementación requiere el ajuste para encontrar los mejores parámetros C y γ, y la escogencia del tipo de función kernel para poder hacer el entrenamiento del clasificador. Para esto se sigue el procedimiento recomendado por Chih-Wei [16] que consiste en:

1. Transformar los datos a formato SVM para lo cual se hacen conversiones en representaciones binarias de las características definidas.

2. Se usa como kernel la función de base radial $K(x, y) = e^{-\gamma\|x-y\|^2}$

3. Se usa búsqueda exhaustiva *(Grid Search)* para encontrar los mejores parámetros C y γ. Se obtienen los valores $C = 32$ y $\gamma = 0,0078125$.

4. Se usan estos parámetros para entrenar en base al conjunto de características obtenidas en el proceso de entrenamiento del analizador al corpus de ANCORA.

A continuación se describe la integración del algoritmo de máquinas de vectores de soporte para clasificación de argumentos de verbo, con el analizador sintáctico probabilístico implementado por Bikel adaptado para el idioma español. Inicialmente se hace una introducción del segundo modelo de Collins y luego se explica la modificación de este modelo usando el clasificador de argumentos de verbo.

3.2. El algoritmo CKY y el modelo 2 de Collins

En el segundo modelo de Collins un árbol sintáctico es representado por m probabilidades de subcategorización, en adición a las n dependencias definidias, las probabilidades de subcategorización corresponden a eventos de la forma *"¿Cúal es la probabilidad de necesitar n complementos de tipo sn al lado derecho?"*. Donde *sn* corresponde al no-terminal de un modificador complemento de verbo. El segundo modelo de Collins se puede describir de la siguiente manera:

- Escoger el núcleo (cabeza) H con probabilidad $P_H(H|P,h)$.
- Escoger los marcos de subcategorización del lado izquierdo y el lado derecho del núcleo, LC y RC con probabilidades $P_{LC}(LC|P,H,h)$ y $P_{RC}(RC|P,H,h)$, cada marco de subcategorización es un multiconjunto especificando los complementos que son requeridos por el núcleo en su lado izquierdo y derecho.
- Generar los modificadores del lado izquierdo y derecho con probabilidades

$$P_l(L_i, l_i|H,P,h,distancia(i-1),LC)$$
$$\text{y}$$
$$P_r(R_i, r_i|H,P,h,distancia(i-1),RC)$$

  Estos requerimientos de marcos de subcategorización son adicionados al contexto condicionado. Cuando los complementos son generados, estos son removidos del multiconjunto de subcategorizaciones requeridas.

En la implementación de Bikel, se ve reflejado en dos partes:

1. En la fase de entrenamiento realiza una identificación de argumentos en complementos y adjuntos a partir de la información que provee ANCORA en los árboles sintácticos. En esta fase se genera un modelo de marcos de subcategorización basado en probabilidades calculadas a partir del corpus. Estas probabilidades son obtenidas en la fase de procesamiento y quedan registradas en el modelo probabilístico.
2. En la fase de análisis o decodificación usa el modelo de marcos de subcategorización creado en el entrenamiento para generar conjuntos de argumentos requeridos cuando intenta unir dos items o subárboles en el algoritmo CKY [27], al momento de unir los ítems, se verifica si un ítem es argumento (complemento/adjunto). Si el ítem es identificado como un argumento se remueve el requerimiento que ha sido adicionado como resultado de calcular su probabilidad basado en el modelo de subcategorización creado en la fase de entrenamiento. para saber si es identificador como argumento se usa el método *isArgumentFast* que debe ser implementado en la clase *Training* del paquete del idioma.

3.3. Modificación del analizador sintáctico

La integración del clasificador de argumentos de verbo con el segundo modelo de Collins puede ser escrita como:

- Sea $\alpha \in N$,
- $COMPLEMENTS(v)$ el conjunto de complementos de v, clasificados por el SVM, donde v es un nodo del tipo grup.verb, por lo tanto $v = H$
- $P_{Comp}(\alpha)$ es una función tal que,
- $P_{Comp}(\alpha) = \begin{cases} 1 & si\, \alpha \in COMPLEMENTS(v) \\ 0 & en\, caso\, contrario \end{cases}$
- Entonces se modifica el algoritmo para que $P_{lc}(LC|P,H,h) = P_{lc}(LC|P,H,h)P_{Comp}(LC)$

La modificación del analizador sintáctico modelo 2 de Collins consiste en la integración del clasificador de argumentos al modelo multilenguaje de Bikel. Se persigue mejorar el rendimiento de este analizador y anotar explícitamente los argumentos de verbo que aparecen en la sentencia de entrada. La modificación se realiza en el algoritmo de Bikel en el cual se identifican los complementos y adjuntos usando el clasificador. Para efectuar este cambio se definen los ítems modificando que hace referencia al ítem del verbo, y el modificador el cual clasifica si es un argumento de verbo complemento o adjunto. Luego al momento de hacer la unión de los dos ítems, si el ítem modificando es un verbo, entonces a los ítems modificadores generados se les aplica la función de extracción de características para clasificar que tipo de complementos son.

$$\overrightarrow{fL_i} = extraer\_caracteristicas(L_i)$$
$$\text{y}$$
$$\overrightarrow{fR_i} = extraer\_caracteristicas(L_i)$$

Donde $\overrightarrow{fL_i}$ y $\overrightarrow{fR_i}$ son los vectores de características lingüisticas binarizadas de los ítems seleccionados bajo el modelo probabilístico L_i y R_i respectivamente. El componente SVM clasifica el ítem con características $\overrightarrow{fL_i}$ y $\overrightarrow{fR_i}$ en uno de estos complementos: (suj, cd, ci, atr, cpred, creg, cag, cc). Se contrasta el tipo de complemento obtenido con los tipos de complementos requeridos por el núcleo según la información del lema del verbo en el corpus. El modelo de Collins se basa en el algoritmo CKY, y en la implementación de Bikel se definen dos operaciones básicas para la construcción de los ítems. Que son *addUnaries* (se crean producciones unarias) y *joinItems* (se crean producciones de binarias).

Bikel también define un método *isArgumentFast* que recibe la etiqueta del ítem para determinar si es o no un argumento. Se adiciona un llamado al método que determina el argumento

a partir del ítem y no de la etiqueta raíz. Este proceso de análisis se modifica de la siguiente manera:

- En la fase de entrenamiento se modificó el componente de entrenamiento de Bikel para generar los archivos de entrenamiento requeridos por el clasificador SVM. Estos archivos de entrenamiento son las características obtenidas de los ítems que hacen parte de los árboles de entrenamiento. ANCORA tiene anotado para los ítems de los árboles aquellos que son identificados como argumentos, y el tipo de argumentos que es (suj, cd, ci, atr, cpred, creg, cag, cc). De esta manera se construyen los archivos de entrenamiento para el SVM.

- Dentro del proceso generativo en la fase de decodificación en el algoritmo CKY cuando se realiza la unión entre dos ítems, y uno de esos ítems es un verbo, se aprovecha el hecho de que ANCORA tiene anotado para cada verbo los argumentos requeridos, se obtienen estos argumentos y son adicionados a un conjunto de argumentos requeridos como lo hace el segundo modelo de Collins. El módulo de clasificación tiene un método que extrae las características lingüisticas del ítem (modificador del ítem con el verbo) y a través de libsvm se determina si es argumento y de serlo, que tipo de argumentos es, si es un adjunto (complementos circunstancial) o si es un complemento (directo, indirectio, etc). Se verifica entonces con el conjunto de argumentos requeridos obtenidos de ANCORA, y en el caso de que no pertenezca al conjunto no se ejecuta la unión de los ítems. En caso contrario, es decir en el que si existe al argumento dentro del conjunto de ANCORA, entonces se permite ejecutar la unión de los ítems.

De esta manera se tiene la idea de que los árboles sintácticos generados cuentan con un mecanismo que proporciona un sentido semántico en cuanto a las relaciones de argumentos con respecto al verbo. El modelo del analizador es generado por el entrenamiento del algoritmo de Bikel bajo el modelo 2 de Collins. Por otra parte las características para el entrenamiento del SVM son extraidas desde ANCORA y se genera el modelo SVM, luego se clasifican los argumentos que alimentan a su vez al analizador pero no influyen en el modelo de Collins. Finalmente, se produce el árbol sintáctico de la sentencia de entrada anotados los argumentos de los verbos. Cabe anotar que Chrupala en [10] define un clasificador de roles semánticos sobre los árboles generados por el algoritmo de Bikel para la asignación de ertiquetas funcionales. Esta clasificación se realiza a la salida producida por el analizador sintáctico, mientras que el clasificador de argumentos de verbo definido en este trabajo es usado junto con el segundo modelo de Collins para producir un árbol sintáctico con mayor precisión. Aunque ambos clasificadores lo hacen sobre árboles generados por el analizador y sobre el corpus de ANCORA, el momento de uso de cada uno es diferente. En las figuras **3-1** y **3-3** se muestran las entradas y salidas correspondientes para las fases de entrenamiento y decodificación.

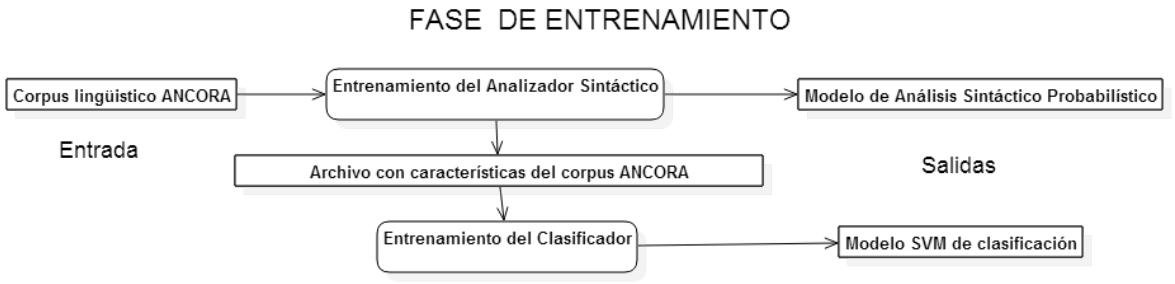

Figura **3-1**: Entradas y salidas de la fase de entrenamiento

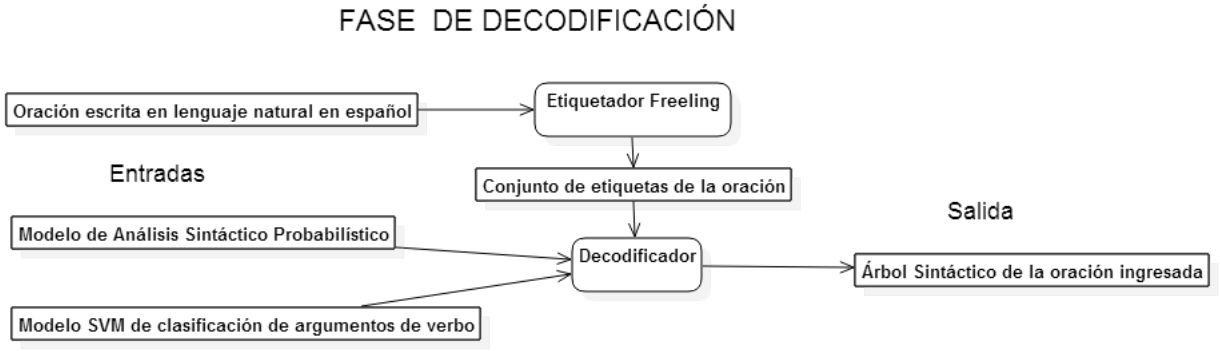

Figura **3-2**: Entradas y salidas de la fase de decodificación

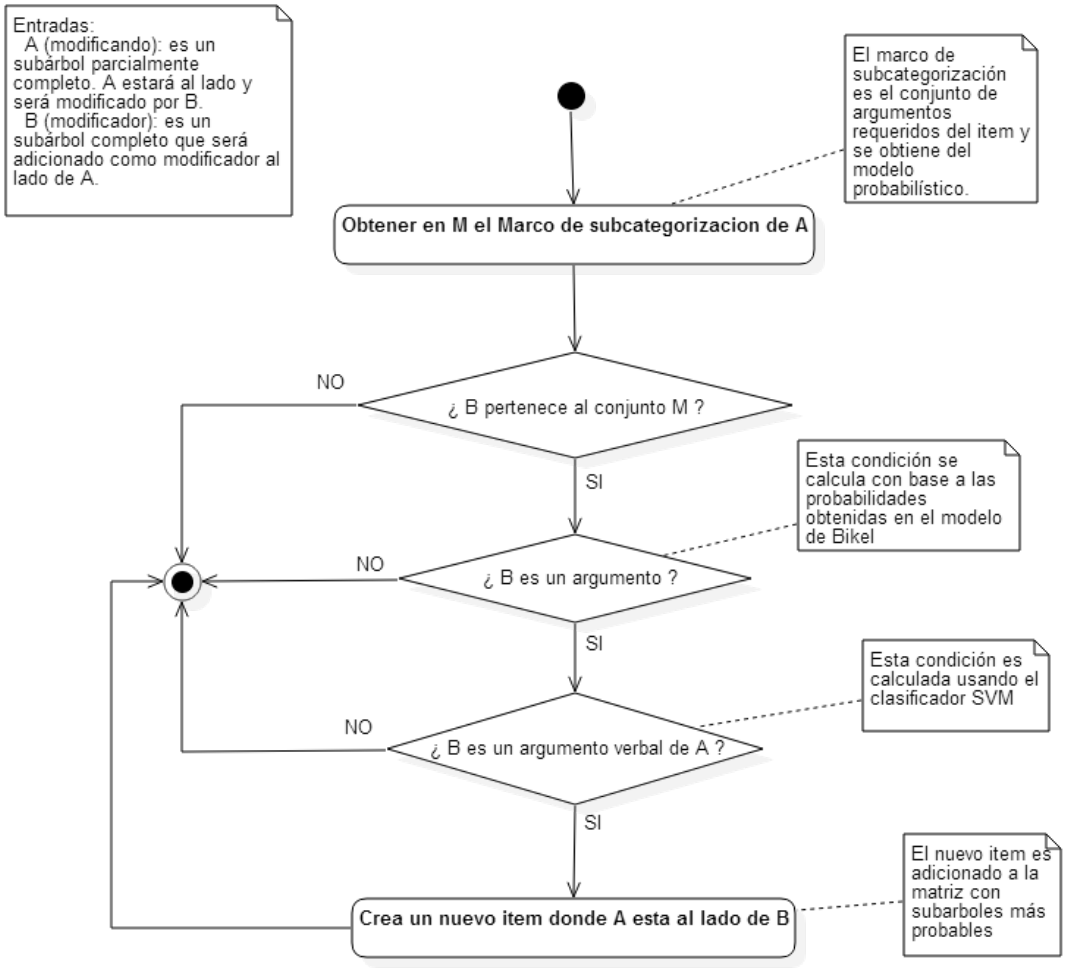

Figura **3-3**: Diagrama de actividades del método *joinItems* modificado.

4 Evaluación de los analizadores sintácticos

En esta sección se presentan los resultados de pruebas realizadas al modelo línea base del analizador sintáctico probabilístico de Bikel-Collins así como los resultados de pruebas al analizador sintáctico probabilístico que usa clasificación de argumentos de verbo. La técnica utilizada para las pruebas es conocida como validación cruzada específicamente dejando uno fuera, la cual consiste en separar los datos de forma que para cada iteración tengamos una sola muestra para los datos de prueba y todo el resto conformando los datos de entrenamiento. Como datos de entrenamiento se usa el ANCORA (CESS-ESP) [1], que cuenta con 610 archivos en formato parentizado (TBF, Treebank Bracketted Format), estos son divididos en 10 particiones de 61 archivos cada uno. A estos archivos tbf se les aplica un algoritmo de extracción con el fin de obtener los textos representados en los árboles con sus respectivas etiquetas, y ser usados posteriormente como archivos de prueba. Una vez obtenidos los árboles generados por el analizador sintáctico se hacen las evaluaciones de comparación con los archivos meta (gold-trees) de cada conjunto y luego se calcula la media aritmética de las f-scores obtenidas para cada una de las pruebas. Para el cálculo del f-score se usa el algoritmo *parseval.py* dado por Kikas y Treumuth en [34] que esta escrito en lenguaje Python. Este algoritmo obtiene las médidas de desempeño (precisión, cobertura y f-score) teniendo en cuenta que P es el árbol analizado automáticamente por el sistema y T el árbol analizado manualmente por lingüístas (gold standard). A continuación se definen las médidas de desempeño:

- Cobertura (Recall): $R = \frac{(\#\ de\ consituyentes\ correctos\ enP)}{(\#\ de\ constituyentes\ correctos\ en\ T)}$
- Precisión: $P = \frac{(\#\ de\ consituyentes\ correctos\ en\ P)}{(\#\ de\ constituyentes\ en\ P)}$
- f-score: Media armónica entre cobertura y precisión f-score $= \frac{(\beta^2+1)*P*R}{\beta^2*P+R}$, donde $\beta = 1$, dado que P y R tienen el mismo peso.

[1] "Conjunto de árboles sintácticos de ANCORA en formato parentizado (TBF, Treebank Bracketted Format)"

4.1. Evaluación del modelo línea base

Para la implementación del modelo línea base se ha implementado el analizador de Bikel adaptado al idioma español. En esta dirección se construye el paquete *javargas.parser.spanish* con las clases *HeadFinder.java*, *WordFeatures.java*, *Treebank.java* y *Training.java* acondicionándolas para trabajar con el corpus ANCORA (CESS-ESP). Para la ejecución de Bikel se redefinen las reglas de cabeza de magerman y se construyen en un lenguaje funcional como Lisp la ruta para encontrar el núcleo (cabeza) de los sintagmas o constituyentes. Las reglas definidas se muestran en la figura **2-7**. Por ejemplo, para encontrar el núcleo de un grupo verbal (grup.verb) el analizador hace el siguiente recorrido; primero busca de derecha a izquierda el primer ítem hijo etiquetado como *infinitu* y este es seleccionado como núcleo. Si no se encuentra entonces se recorre de derecha a izquierda el primer ítem *gerundi*, de no ser encontrarlo se busca de derecha a izquierda el primer ítem *vmp* y así hasta completar las demas reglas de cabeza para *grup.verb*. Cabe resaltar que las 27 reglas de cabeza de la figura **2-7** están basadas en la funcionalidad sintáctica definida en ANCORA. Este es un trabajo complejo porque se deben tener todas las posibilidades para encontrar los núcleos en cada una de las funciones sintácticas de ANCORA. De esta manera este conjunto de reglas de cabeza se han ido estructurando de acuerdo a las pruebas realizadas. Físicamente, las reglas de cabeza se guardan en un archivo *head-rules.lisp* las cual son ejecutadas por el analizador de Bikel indicando en los parámetros de ejecución los archivos de configuración ubicado en el archivo *settings/spanish.properties*. En este último archivo se le indica al parser de Bikel usar el paquete construido para el idioma español.

4.1.1. Análisis de resultados

Para la evaluación del clasificador de línea base fue utilizado, un conjunto de entrenamiento de 4.000 árboles sintácticos y un conjunto de testeo de 370 árboles particionados en 10 conjuntos sobre todo el corpus de ANCORA. Los árboles generados por el analizador son comparados con los árboles del corpus que han sido seleccionados como árboles meta (gold trees). Para la comparación de estos árboles, se usa el algoritmo parseval para obtener la precisión y cobertura de los árboles generados por el analizador sintáctico. En general, el desempeño del clasificador de Bikel sobre ANCORA tiene una precisión del 63 %, cobertura del 62 % con un f-score del 63 % como lo muestra la tabla **4-4**. Cabe anotar que se toma como punto de referencia las oraciones de menos de 40 palabras (Ver tabla **4-3**). En este sentido, se puede explicar que para conjuntos de oraciones de más de 40 palabras la precisión sufre una baja debido a que el número de constituyentes de T (árbol analizado manualmente por lingüístas), es mucho mayor que el número de constituyentes correctos en P (árbol analizado automáticamente por el sistema). De igual forma la cobertura aumenta porque el número de constituyentes correctos en T disminuye con relación al número de constituyentes correctos

Conjunto	Presición	Cobertura	F1-score
1	43 %	52 %	47 %
2	70 %	72 %	71 %
3	84 %	82 %	83 %
4	83 %	82 %	82 %
5	37 %	53 %	44 %
6	65 %	68 %	67 %
7	58 %	58 %	58 %
8	42 %	45 %	43 %
9	39 %	77 %	52 %
10	84 %	82 %	83 %

Tabla **4-1**: Precisión del analizador sintáctico línea base de Bikel.

en P. Por lo tanto, se puede deducir que para oraciones mayores de 40 palabras el analizador debe inferir un conjunto mayor de reglas de cabeza. Esto también porque la estructura interna del analizador cambia en el sentido de los adjuntos y complementos, pero no se ve reflejado en la anotación explícita. Comparando estos resultados con los realizados en trabajos anteriores como el de Chrupala [10] se puede observar que el f-score baja, primero porque el conjunto de reglas de cabeza es menor y también porque la estructura que modifica el modelo 2 de Collins es una estrucutra LFG en la cual la forma de encontrar el núcleo es diferente a las reglas de Magerman [22] usadas en este trabajo. En esta dirección Chrupala define un corpus y un conjunto de reglas para la generación de los árboles sintácticos. Siendo una diferencia marcada con uno de los objetivos de este trabajo que era el de anotar explícitamente los adjuntos y complementos.

4.2. Ajustes de ejemplos para el clasificador de argumentos de verbo

En el proceso de entrenamiento del analizador sintáctico probabilístico, se generan los archivos de características extraidas de los árboles del corpus ANCORA y que son usados como datos de entrenamiento del clasificador de argumentos de verbo. Para el clasificador de argumentos de verbo no se realizó una evaluación de validación cruzada, por lo que no es un objetivo del trabajo de investigación, pero si se realiza una precisión relativa usando la curva ROC (Receiver Operating Characteristic; en adelante Área bajo la curva) que representa el equilibrio entre los datos positivos y negativos para el entrenamiento del clasificador. Para este ajuste se usó la herramienta plotroc.py que da el área bajo la curva usada para presentar

resultados en problemas de decisión binaria en máquinas de aprendizaje como se presenta en [5]. Se calcula la precisión según los datos de entrenamiento obteniendo los resultados mostrados en la tabla **4-2**. Las gráficas generadas por el plotroc.py a través de gnuplot son mostradas en la figura **4-1**. La implementación del SVM se hace usando la librería libsvm [7] usando como parámetros un $C = 32$ y $\gamma = 0,0078125$ que fueron obtenidos a través del método de búsqueda exhaustiva, con la herramienta grid.py provista en el sitio de libsvm, para un kernel de función de base radial $K(x, y) = e^{-\gamma\|x-y\|^2}$.

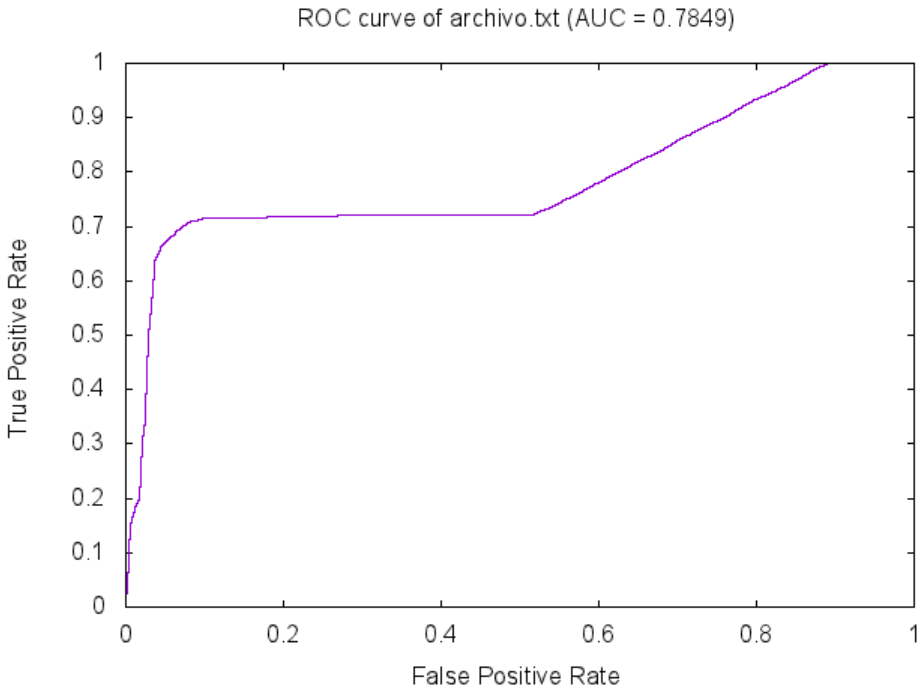

Figura **4-1**: Gráfica generada por el gnuplot de los archivos de los ejemplos del SVM.

4.3. Evaluación del analizador sintáctico con clasificación de argumentos de verbo

Para la evaluación del analizador sintáctico probabilístico con el clasificador de argumentos de verbo, se realiza el mismo procedimiento usado para la evaluación del analizador sintáctico línea base de Bikel. La evaluación es sobre la modificación del algoritmo en la clase *danbikel.parser.Decoder* en el método *joinItems* del analizador de Bikel para que en el momento de unir dos items si uno de los items es un constituyente verbo, se ejecute la clasificación binaria de argumento de verbo al item modificador del constituyente verbal.

Conjunto	Valor de plotroc
1	Accuracy = 81.075 % (3243/4000) (classification) Accuracy = 82.125 % (3285/4000) (classification) Accuracy = 81.05 % (3242/4000) (classification) Accuracy = 81.725 % (3269/4000) (classification) Accuracy = 81.425 % (3257/4000) (classification)
2	Accuracy = 83.5 % (3340/4000) (classification) Accuracy = 83.725 % (3349/4000) (classification) Accuracy = 82.1 % (3284/4000) (classification) Accuracy = 81.1 % (3244/4000) (classification) Accuracy = 81.4046 % (3257/4001) (classification)
3	Accuracy = 80.2451 % (3209/3999) (classification) Accuracy = 80.275 % (3211/4000) (classification) Accuracy = 81.225 % (3249/4000) (classification) Accuracy = 80.2 % (3208/4000) (classification) Accuracy = 80.325 % (3213/4000) (classification)
4	Accuracy = 81.4 % (3256/4000) (classification) Accuracy = 82.05 % (3282/4000) (classification) Accuracy = 81.275 % (3251/4000) (classification) Accuracy = 81.825 % (3273/4000) (classification) Accuracy = 82.1 % (3284/4000) (classification)

Tabla **4-2**: Precisión relativa de los ejemplos de la máquina de soporte vectorial usando el área bajo la curva.

Modelo	Constantes Etiquetadas			
	<= 40 palabras		<= 70 palabras	
	Precisión	Cobertura	Precisión	Cobertura
Linea base	63 %	62 %	37 %	69 %
SVM	63 %	62 %	37 %	69 %

Tabla **4-3**: Precisión del analizador línea base y analizador con SVM

	Presición	Cobertura	Medida F
Chrupala Baseline	73,95	70,67	72,27
Chrupala SVM	76,90	74,48	75,67
Baseline	63,33	62,38	62,83
SVM	63,33	62,38	62,83

Tabla **4-4**: Evaluación de resultados del analizador Baseline y SVM

4.3.1. Análisis de resultados

Al igual que para el entrenamiento del analizador de Bikel se definió un conjunto de entrenamiento de 4.000 árboles sintácticos y un conjunto de pruebas de 370 árboles particionados en 10 conjuntos sobre todo el corpus de ANCORA. En general, la precisión del analizador fue la misma que la del modelo de línea base porque el trabajo estuvo enfocado en la consolidación del modelo 2 Collins para la generación de estructuras sintácticas. En este sentido, se comprueba que la precisión del modelo base de Bikel y el modelo con clasificador de argumentos tienen la misma precisión y cobertura sobre el mismo conjunto de entrenamiento. Con respecto al mejoramiento del analizador base se implementó una estructura que permite la anotación explícita de los adjuntos y complementos pero que no fue evaluada. Lo anterior quiere decir que se fija una estructura sintáctica-semántica sin todavía probar pero que tiene una precisión igual al modelo base (Bikel). El clasificador de argumentos es un aporte al trabajo realizado sobre analizadores sintácticos porque provee una distinción entre sus argumentos verbales de complementos y adjuntos, permitiendo extraer una estructura básica elimando los adjuntos y sin perder el sentido la oración, y provee al analizador de una estructura semántica de características lo cual no garantiza inicialmente el mejoramiento de los analizadores, se espera que en otras pruebas con la anotación explícita de los complementos y adjuntos se puedan evidenciar muestras de mejoramiento del modelo de Bikel.

5 Conclusiones y recomendaciones

5.1. Conclusiones

En este trabajo de investigación se integró en el analizador sintáctico probabilístico de Bikel un clasificador de argumentos de verbo para anotar explícitamente la distinción entre complementos y adjuntos, esta distinción permite extraer un subárbol de estructura básica de una oración sin perder ni la esencia ni el sentido de la oración, elimando del árbol sintáctico los constituyentes clasificados como argumentos verbales de adjunción. Por otro lado se estudió la posibilidad de mejorar la precisión del analizador sintáctico modificando en el algoritmo de Bikel, el segundo modelo de Collins, el cuál se basa en identificar argumentos para eliminar los requeridos dado los marcos de subcategorización que son obtenidos a través de un tratamiento probabilístico a partir del corpus ANCORA. Aunque tener más efectividad en esta identificación supone una mejora en la precisión del análisis sintáctico, se encontró que esta precisión se mantiene igual, lo que significa que el tratamiento probabilístico de Collins no mejora con el clasificador de argumentos incorporado en su segundo modelo. Como resultado se obtiene:

- Un analizador sintáctico probabilístico para el idioma español entrenado con el corpus de ANCORA.
- Un clasificador de argumentos de verbo para el idioma español con máquinas de vectores de soporte entrenado con información del corpus de ANCORA.
- La integración del clasificador de argumentos de verbo como lo hace el segundo modelo de Collins en la implementación de Bikel.

5.2. Trabajo Futuro

- Aprovechar la información suministrada por el corpus Ancora sobre los argumentos de los verbos, construyendo marcos de subcategorización más precisos.
- Definir más características léxicas, sintácticas, semánticas y contextuales para mejorar el componente clasificador basado en máquinas de vectores de soporte.
- Optimizar el proceso de extracción de características.

- Utilizar técnicas de validación cruzada para encontrar un mejor ajuste en los parámetros C y γ del clasificador de argumentos de verbo.
- Definir un conjunto de pruebas con la anotación explícita de adjuntos y complementos de verbos.

Bibliografía

[1] *The Penn Treebank Project*. 1999

[2] Allen, James: *Natural Language Understanding (2Nd Ed.)*. Redwood City, CA, USA : Benjamin-Cummings Publishing Co., Inc., 1995. – ISBN 0–8053–0334–0

[3] Allen, James: *Natural Language Understanding (2Nd Ed.)*. Redwood City, CA, USA : Benjamin-Cummings Publishing Co., Inc., 1995. – ISBN 0–8053–0334–0

[4] Bikel, Daniel M.: *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Philadelphia, PA, USA, Tesis de Grado, 2004. – AAI3152016

[5] Bradley, Andrew P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. En: *Pattern Recognition* 30 (1997), p. 1145–1159

[6] Carreras, Xavier ; Chao, Isaac ; Padró, Lluís ; Padró, Muntsa: FreeLing: An Open-Source Suite of Language Analyzers. En: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, 2004

[7] Chang, Chih-Chung ; Lin, Chih-Jen: LIBSVM: A Library for Support Vector Machines. En: *ACM Trans. Intell. Syst. Technol.* 2 (2011), Mai, Nr. 3, p. 27:1–27:27. – ISSN 2157–6904

[8] Charniak, Eugene: Statistical Parsing with a Context-free Grammar and Word Statistics. En: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, 1997 (AAAI'97/IAAI'97). – ISBN 0–262–51095–2, p. 598–603

[9] Chomsky, Noam: Syntactic Structure. 1957

[10] Chrupala, Grzegorz ; van Genabith, Josef: Using Machine-learning to Assign Function Labels to Parser Output for Spanish. En: *Proceedings of the COLING/ACL on Main Conference Poster Sessions*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2006 (COLING-ACL '06), p. 136–143

[11] Collins, Michael: Three Generative, Lexicalised Models for Statistical Parsing. En: *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1997 (EACL '97), p. 16–23

[12] Collins, Michael: Head-Driven Statistical Models for Natural Language Parsing. En: *Comput. Linguist.* 29 (2003), Dezember, Nr. 4, p. 589–637. – ISSN 0891–2017

[13] Cortes, Corinna ; Vapnik, Vladimir: Support-Vector Networks. En: *Mach. Learn.* 20 (1995), September, Nr. 3, p. 273–297. – ISSN 0885–6125

[14] Cowan, Brooke A.: *A Tree-to-Tree Model for Statistical Machine Translation.* Massachusetts, MA, USA, Tesis de Grado, 2008

[15] Gorn, Saul. *Explicit definitions and linguistic dominoes. Systems and Computer Science.* 1967

[16] wei Hsu, Chih ; chung Chang, Chih ; jen Lin, Chih. *A practical guide to support vector classification.* 2010

[17] Joshi, Aravind K. ; Schabes, Yves: Handbook of Formal Languages, Vol. 3. New York, NY, USA : Springer-Verlag New York, Inc., 1997. – ISBN 3–540–60649–1, Kapitel Tree-adjoining Grammars, p. 69–123

[18] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* 1st. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2000. – ISBN 0130950696

[19] di Linguistica Computazionale, Istituto. *Expert Advisory Group for Language Engineering Standards.* 2004

[20] Llorach, E.A.: *Gramática de la lengua española.* Espasa Calpe, 1994 (Colección Nebrija y Bello). – ISBN 9788423978403

[21] Madrigal, Víctor J. Díaz: Gramáticas de Adjunción de Árboles: Un Enfoque Deductivo en el Análisis Sintáctico. En: *Procesamiento del Lenguaje Natural* 28 (2002), Nr. 0. – ISSN 1989–7553

[22] Magerman, David M.: Statistical Decision-Tree Models for Parsing. En: *In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995, p. 276–283

[23] Marco, Antonio M.: *Desambiguación en procesamiento del lenguaje natural mediante técnicas de aprendizaje automático*, Tesis de Grado, 2004

[24] Mariona Taulé, M. Antónia M. ; Recasens, Marta: AnCora: Multilevel Annotated Corpora for Catalan and Spanish. En: Nicoletta Calzolari (Conference Chair), Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel T. (Ed.): *Proceedings of the Sixth International Conference on Language Resources and*

Evaluation (LREC'08). Marrakech, Morocco : European Language Resources Association (ELRA), may 2008. – http://www.lrec-conf.org/proceedings/lrec2008/. – ISBN 2–9517408–4–0

[25] Martí, M.A. ; Taulé, M. ; Bertran, M. ; Márquez, L. *AnCora: Multilingual and Multilevel Annotated Corpora*. 2007

[26] Müller, H.H. ; d'Etudes Romanes], [Université Copenhague]. [.: *Los adjuntos como componentes del sintagma nominal*. 2000

[27] Oncina, Jose. *The Cocke-Younger-Kasami algorithm for cyclic strings*

[28] Padró, Lluís ; Stanilovsky, Evgeny: FreeLing 3.0: Towards Wider Multilinguality. En: *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. Istanbul, Turkey, May 2012

[29] Pradhan, Sameer ; Hacioglu, Kadri ; Krugler, Valerie ; Ward, Wayne ; Martin, James H. ; Jurafsky, Daniel: Support Vector Learning for Semantic Argument Classification. En: *Mach. Learn.* 60 (2005), September, Nr. 1-3, p. 11–39. – ISSN 0885–6125

[30] Pradhan, Sameer ; Ward, Wayne ; Hacioglu, Kadri ; Martin, James H.: Shallow semantic parsing using Support Vector Machines, 2004

[31] Reyes, José A. ; Montes, Azucena ; González, Juan G. ; Pinto, David E.: Clasificación de roles semánticos usando características sintácticas, semánticas y contextuales. En: *Computación y Sistemas* 17 (2013), Nr. 2

[32] Reyes, José A. ; Montes, Azucena ; González, Juan G. ; Pinto, David E.: Clasificación de roles semánticos usando características sintácticas, semánticas y contextuales. En: *Computación y Sistemas* 17 (2013), Nr. 2

[33] Sameer Pradhan, Wayne W. ; Hacioglu, Kadri ; Martin, James H.: Dan Jurafsky. En: *Shallow Semantic Parsing using Support Vector Machines*

[34] Taavet Kikas, Margus T. *Automatic Parser Evaluation*. 2007

[35] Tesnière, L ; Klincksieck, Editions (Ed.): *Elements de syntaxe structurale*. Editions Klincksieck, 1959

[36] Vapnik, V. N.: An Overview of Statistical Learning Theory. En: *Trans. Neur. Netw.* 10 (1999), September, Nr. 5, p. 988–999. – ISSN 1045–9227

[37] y Vicente Carrillo Montero y Víctor J. Díaz Madrigal, Miguel A. Alonso P.: Análisis sintáctico combinado de gramáticas de adjunción de árboles y de gramáticas de inserción de árboles. En: *Procesamiento del Lenguaje Natural* 29 (2002), Nr. 0. – ISSN 1989–7553

[38] Vilares, Miguel A. Alonso Carlos Gómez J.: Análisis Sintáctico, COLE Research Group, 2010. – Web; accedido el 09 Noviembre de 2014