Universidad Nacional de Córdoba
Facultad de Matemática Astronomía y Física

Trabajo Especial

Inferencia de Estructuras Gramaticales Mediante Descomposiciones a Estructuras Simples.

Alejandro Javier Peralta Frias
Director: Gabriel Infante-Lopez

---

03 de agosto de 2007

Resumen

En el presente trabajo especial desarrollamos técnicas para inferir "etiquetas complejas" a partir de poco material de entrenamiento. Nuestra idea consiste en separar las etiquetas en "partes significativas" y entrenar *taggers* para que infieran estas "partes significativas" a partir de una secuencia de palabras. De la secuencia resultante de partes inferidas recuperamos las etiquetas originales. Nuestra hipótesis es que la descomposición de las etiquetas genera mayor regularidad en los datos y al dividir las etiquetas en dos o más conjuntos, reducimos el conjunto de símbolos con el cual entrenamos el *tagger* y de esta manera disminuimos los problemas causados por la escasez de datos. Probamos nuestras técnicas en la tarea especifica de *POS tagging*.

Palabras clave: part-of-speech tagging, chunking, etiquetas, descomposición, material de entrenamiento, escasez de datos.

Clasificación:

I.2.6 Learning

I.2.7 Natural Language Processing

Agradecimientos

A mis padres por el apoyo que me dieron.

A mis hermanos Guille, Ceci y Rodri por bancarse un hermano mayor como yo.

A Carlita... por todo!

Al Tavo, Fede, Nati, Fer, Tucu, Santi por el aguante...

A Gabi ("el dire") por la buena onda.

Al Grupo de PLN: Laura, Martincito, Franco, Martín, Sergio por las charlas, las sugerencias y el café!

A todos mis amigos y compañeros con quienes compartí alegrías, penas, copas, morfi, luchas, joda, trabajo y horas de estudio (ja!)

Índice general

1. Introducción **1**

2. Combinación de Lenguajes **7**
2.1. Formalización 7
2.1.1. Un Ejemplo 8
2.2. Implementación de las Funciones 10

3. Taggers **11**
3.1. Introducción 11
3.2. Fuentes de Información 13
3.3. Máxima Entropía 14
3.3.1. Representando Evidencia 15
3.3.2. Probando el Modelo 18
3.4. Otros Métodos 19

4. Chunking **21**
4.1. Introducción 21
4.2. Definiendo *Chunks* 21
4.3. CONLL 22
4.3.1. Material de Entrenamiento 23
4.3.2. *Chunking* tratado como *POS Tagging* 23

5. Experimentos **27**
5.1. Introducción 27
5.2. Procedimiento General 28
5.3. Experimentando con Material Artificial 31
5.3.1. Generando Etiquetas Complejas 32
5.3.2. Generando Etiquetas Simples 34
5.3.3. Métodos de Combinación 36
5.3.4. Resultados 38
5.3.5. Reduciendo la Cantidad de Material 40

5.4. Cast3LB 42

6. Conclusiones **47**

Capítulo 1

Introducción

El Procesamiento de Lenguaje Natural (PLN) es una subrama de la inteligencia artificial y de la lingüística. El PLN se ocupa de los problemas de crear representaciones formales del lenguaje hablado y/o escrito por humanos (*lenguaje natural*) para que la manipulación automática del mismo sea más fácil. El objetivo del PLN es poder construir mecanismos y sistemas computacionales que permitan la comunicación entre las personas y la máquinas por medio del lenguaje natural. En otras palabras, se busca poder crear programas que puedan analizar, comprender y generar lenguaje natural de manera que los usuarios puedan llegar a interactuar con las computadoras a través de lenguaje humano. Además, el PLN se aplica en el campo del procesamiento y la recuperación de la información, en la traducción automática, el reconocimiento del habla y la corrección ortográfica de textos.

Dentro del PLN es común tratar a los problemas como tareas de clasificación, y en particular, de clasificación estadística: esto es, dado un conjunto de objetos lingüísticos y el contexto en donde éstos ocurren, determinar a qué clase pertenece cada objeto según un modelo probabilístico $p(a, b)$ que fija la probabilidad de encontrar el objeto lingüístico a en el contexto b. Lo que se considera por contexto depende de la aplicación, aunque es muy común que los contextos incluyan palabras. En algunas aplicaciones el contexto es una sola palabra, mientras en que otras, el contexto son varias palabras con algunas etiquetas asociadas. Algunos ejemplos de aplicaciones de PLN vistas como problemas de clasificación se describen a continuación:

POS Tagging es el proceso de asignar etiquetas a palabras de un texto según su *part-of-speech* (categoría morfosintáctica). La categoría morfosintáctica hace referencia al rol que cumple la palabra dentro de una oración (verbos, nombres, adjetivos, etc.). Esta aplicación tiene como entrada una oración y como salida cada palabra etiquetada con la cate-

goría a la que ésta puede pertenecer en la oración. El contexto en esta aplicación está dado por las las palabras vecinas en la oración. Es importante notar que las palabras se pueden clasificar en distintas categorías y por lo tanto, la clasificación involucra no sólo un proceso de selección sino también un proceso de desambiguación. La información que provee esta aplicación es útil para tareas asociadas al procesamiento de la información como la Extracción de la Información, Recuperación de la Información y en el Análisis Sintáctico, entre otras tareas. El *Tagging* provee una capa de representación útil y accesible.

Chunking consiste en dividir un texto en partes que se corresponden a conjuntos de palabras sintácticamente correlacionadas, los constituyentes de la oración; a cada una de éstas partes la llamamos un *Chunk*. El *Chunking* también puede ser visto como un análisis de oraciones que identifica los constituyentes sin buscar otra estructura interna. Los *chunks* no son estructuras recursivas y tampoco se sobreponen, esto quiere decir que una palabra, en una oración, puede pertenecer solo a un *chunk*. Dado que los tipos de *Chunk* están fijos, esta tarea se puede tratar como una de clasificación. Dada una oración, un *Chunker* produce una secuencia de anotaciones correspondientes a cada palabra de la oración. Las etiquetas identifican en qué posición del *Chunk* se encuentra cada palabra y el tipo de *Chunk* en la que se encuentra. El *Chunking* también se utiliza en la Extracción de la Información y puede proveer de datos útiles (*features*) para la tarea de Reconocimiento de Nombres de Entidades (Named Entity Recognition), entre otras cosas.

Word Sense Desambiguation (*Desambiguación de Sentidos*) es la tarea de determinar cúal de los sentidos de una palabra ambigua (con más de un sentido) es pertinente en una oración. Por ejemplo la palabra "*banco*" puede tener el sentido de asiento o de entidad financiera y, el sentido informal ilustrado por la frase "*Yo me banco esto!*". Con este último ejemplo queremos decir que pueden existir otros sentidos, más allá de los contemplados en el diccionario. Sin embargo, en la práctica, por lo general, se fija la cantidad de sentidos que puede tener una palabra. La entrada de esta tarea es una "ventana" de texto alrededor de la palabra (contexto) que deseamos desambiguar y la salida es la palabra asociada a una "etiqueta semántica". Esta aplicación es muy útil para la Traducción Automática.

Las tareas anteriormente descriptas son "intermedias", es decir que no constituyen un objetivo en sí, pero son necesarias en algún nivel para superar la mayoría de los problemas en el PLN, como Traducción Automática,

Análisis Sintáctico, Recuperación de la Información, etc. Las tareas de clasificación se resuelven utilizando métodos supervisados de aprendizaje como por ejemplo: HMM, Máxima Entropía, Naïve Bayes, Support Vector Machines, entre otros. Los métodos supervisados se caracterizan por requerir material de aprendizaje que en general se compone de secuencias de pares (palabra, etiqueta). Las etiquetas en el material conceptualizan la "clase" del objeto lingüístico a la cual la palabra etiqueta pertenece mientras que en general, las palabras son el objeto lingüístico. A continuación presentamos dos ejemplos de material de entrenamiento.

- En *POS Tagging* se utilizan oraciones donde cada palabra está anotada con su correspondiente *POS Tag*. Aquí vemos una oración en Inglés, luego de cada palabra y separado por un guión bajo se encuentra su respectivo *POS Tag*. Las etiquetas identifican entre otras categorías a: pronombre personal (`PRP`), verbo – presente, 3º persona, singular – (`VBZ`), determinantes (`DT`), adjetivos (`JJ`) , nombres en singular (`NN`), etc.

```
He_PRP reckons_VBZ the_DT current_JJ account_NN
deficit_NN will_MD narrow_VB to_TO only_RB
$_$ 1.8_CD billon_NN in_IN September_NNP ._.
```

- Como explicamos previamente, las etiquetas utilizadas en *Chunking* consisten de dos partes, una que indica la posición de la palabra en el *Chunk* y otra parte de la etiqueta que indica el tipo del *chunk*. La letra `B` indica inicio del *Chunk*, `I` indica el medio mientras que `NP` y `VP` son los tipos de *Chunk* que representan frase nominal y frase verbal respectivamente. Más abajo vemos la misma oración en donde delimitamos los *Chunks* con corchetes.

```
He          B-NP
reckons     B-VP
the         B-NP
current     I-NP
account     I-NP
deficit     I-NP
will        B-VP
narrow      I-VP
...
```

$[_{NP}$ He $][_{VP}$ reckons $][_{NP}$ the current account deficit $]$
$[_{VP}$ will narrow $][_{PP}$ to $][_{NP}$ only \$ 1.8 billon $]$
$[_{PP}$ in $][_{NP}$ September $]$

Dependiendo de la aplicación, las etiquetas pueden ser símbolos simples (como por ejemplo en *POS tagging*) o compuestos o complejos (como por ejemplo en *Chunking*). Como los algoritmos de *tagging* necesitan de material de entrenamiento, cuanta mayor complejidad tengan las etiquetas, mayor será el tamaño del material de entrenamiento que se necesita, y aun, cuando contamos con grandes cantidades de material de entrenamiento, puede suceder que faltan datos útiles para clasificar cierto objeto lingüístico. A este problema se lo conoce como el problema de escasez de datos (*sparse data problem*). La creación de material de entrenamiento es costosa y son necesarias técnicas que intenten inducir etiquetas complejas de poco material de entrenamiento.

En este trabajo abordamos este último punto y proponemos solucionar este problema de la siguiente manera:

I. En primera instancia descomponemos las etiquetas complejas en etiquetas simples,

II. luego se entrenan *taggers* para la recuperación de las etiquetas simples,

III. y finalmente se recomponen las etiquetas complejas a partir de las simples.

Nuestra idea consiste en separar las etiquetas en "partes significativas" que puedan reemplazar, en nuevos materiales de entrenamiento, a las etiquetas complejas y así entrenar *taggers* que infieran estas "partes significativas" de las etiquetas dada una secuencia de palabras. A partir de la secuencia de partes inferidas recuperamos las etiquetas originales. Suponemos que este método es superior en eficacia a entrenar un *tagger* con las etiquetas complejas directamente. Para ilustrar nuestra idea, veamos como ejemplo nuestra propuesta aplicada a las etiquetas que se utilizan para denotar las diferentes conjugaciones verbales en *POS Tagging*, siendo estas las siguientes: `VBN`, `VBG`, `VBP`, `VBD`, `VBZ`, `VB`[1].Descomponemos cada etiqueta en dos partes `VB - N`, `VB - G`, `VB - P`, `VB - D`, `VB - Z`, `VB` - ϵ, donde ϵ representa ausencia de las etiquetas `N, G, P, D, Z`. De esta manera "generalizamos" la etiqueta `VB`. A partir

[1] `VB` verb, base form, e.g. take - `VBD` verb, past tense, e.g. took - `VBG` verb, gerund/present participle, e.g. taking - `VBN` verb, past participle, e.g. taken - `VBP` verb, sing. present, non-3d, e.g. take - `VBZ` verb, 3rd person sing. present, e.g. takes

de esta descomposición entrenamos un *tagger* que decida si una palabra es un verbo y entrenamos otro *tagger* que infiera la etiqueta correspondiente del conjunto {`N, G, P, D, Z,` ϵ}, determinando así la conjugación verbal exacta.

Nuestra hipótesis es que la descomposición de las etiquetas genera mayor regularidad en los datos. Existen más secuencias en el material de entrenamiento que tienen el mismo patrón. Como vimos en el ejemplo anterior, generalizamos un conjunto de etiquetas a solamente una, `VB`. Además al dividir las etiquetas en dos o más conjuntos, reducimos el conjunto de símbolos con el cual entrenamos el *tagger* y de esta manera disminuimos los problemas causados por la escasez de datos. A partir de esta evidencia exploraremos la eficacia de nuestra propuesta y veremos si hay mejora significativa respecto los métodos convencionales para etiquetar. Los objetivos pueden resumirse en los siguientes puntos:

I. ¿Qué formas de descomposición existen?

II. ¿Qué ventajas tendría esto con respecto al método original?

III. ¿Los métodos de recomposición podrían agregar errores?

Para desarrollar este procedimiento suponemos que contamos con material de entrenamiento que contenga al conjunto de etiquetas con el cual estamos trabajando y que las etiquetas se comportan de manera similar a los *POS Tags*. Es decir que reúnen las siguientes características:

- Cada palabra está asociada a una o más etiquetas.
- Las etiquetas de cada palabra, en una oración o frase, quedan unívocamente determinadas.
- La etiqueta que le corresponde a cada palabra en una oración está determinada por el contexto en el cual la palabra se encuentra.

En los resultados que obtuvimos, observamos que las descomposiciones son particulares para cada conjunto de etiquetas y dependen de la estructura de las etiquetas. Los métodos de descomposición pueden generar pérdida de información sobre la relación entre las partes que componen a la etiqueta, quizás útil a la hora de inferir las etiquetas complejas. Implementamos métodos simples para la recomposición de las etiquetas y utilizándolos no alcanzamos los valores de referencias propuestos (la precisión de los *taggers*

entrenados con etiquetas complejas). Por último, el único tipo de error que podemos detectar es la invalidez de las etiquetas al combinarlas. Los errores que cometen los *taggers* entrenados con etiquetas simples se propagan a la composición dado que no se pueden corregir ya que no contamos con mayor información de la que nos provee los *taggers*.

El trabajo se desarrollará de esta manera; en el capítulo 2 introducimos la metodología empleada. La descripción de *Tagging* y los modelos de Máxima Entropía para desambiguar *POS Tags* serán abordados en el capítulo 3, y en el 4 definimos *Chunking* y describimos un método que utiliza *POS Taggers*. En el 5 presentamos los experimentos que hicimos a partir de una combinación de etiquetas de *Chunking* y *POS Tags* utilizando *taggers* para desambiguar. Las conclusiones se encuentran en el capítulo 6.

Capítulo 2

Combinación de Lenguajes

2.1. Formalización

En este capítulo procuramos dar una forma general y formal de nuestro procedimiento. Partimos de la suposición de que contamos con material de entrenamiento que consiste en un conjunto de secuencias de pares $(palabra, etiqueta)$, donde cada etiqueta pertenece a Σ. En nuestro contexto particular, cada secuencia es una oración o frase cuyas palabras están anotadas con etiquetas de Σ. Además contamos con los siguientes conjuntos de etiquetas $\Sigma_1, \ldots, \Sigma_k$ y funciones $f_1, \ldots, f_k : \Sigma \rightarrow \Sigma_k$ tales que $\forall \alpha \in \Sigma : f_i(\alpha) \in \Sigma_i$ y la función $f : \Sigma_1 \times \ldots \times \Sigma_k \rightarrow \Sigma$. Las funciones f_i representan un método de descomposición o de proyección de las etiquetas mientras que f recompone las etiquetas $\alpha \in \Sigma$ a partir de elementos de los conjuntos Σ_i con $i \in 1 \ldots k$.

Dada una secuencia $(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_j, \alpha_j)$ donde p_i son palabras y $\alpha_k \in \Sigma$, generamos la secuencias $(p_1, f_i(\alpha_1)), p_2, f_i(\alpha_2)), \ldots, (p_j, f_i(\alpha_j))$ para $i = 1, \ldots, k$. Aplicando esta transformación a todo el material de entrenamiento original, generamos k materiales para entrenar *taggers* que nos permitan inferir etiquetas de Σ_i. Además, podríamos crear material que nos permita inferir etiquetas a partir de otro conjunto de etiquetas. Utilizando las secuencias $(p_1, f_1(\alpha_1)) \ldots (p_j, f_1(\alpha_j))$ y $(p_1, f_2(\alpha_1)) \ldots (p_j, f_2(\alpha_j))$ formamos la secuencia $(f_1(\alpha_1), f_2(\alpha_1)) \ldots (f_1(\alpha_j), f_2(\alpha_j))$.

Utilizando *POS Taggers* induciremos modelos que nos permitan etiquetar palabras con etiquetas de los conjuntos $\Sigma_1 \ldots \Sigma_k$. Luego, para recuperar la etiqueta original de Σ, de alguna palabra, usamos alguno de los siguientes métodos:

I. Inferimos las etiquetas $\Sigma_1 \ldots \Sigma_k$ y aplicamos la función f.

II. Inducimos las etiquetas de Σ_1 y usando las etiquetas de Σ_1 inferimos Σ_2. Luego podemos repetir el procedimiento para los conjuntos que siguen, infiriendo Σ_{j+1} a partir de Σ_j. Dicho mal y pronto, estamos creando una "cascada" de *POS Taggers*.

Suponemos, como lo adelantamos en el capítulo introductorio, que el conjunto Σ se comporta de manera similar a los *POS Tags*, o sea, es suficiente un contexto local para inferir adecuadamente la etiqueta correspondiente.

2.1.1. Un Ejemplo

Para ilustrar nuestra formalización daremos un ejemplo donde suponemos que nuestras etiquetas complejas son pares ordenados. Imaginemos que contamos con material de entrenamiento compuesto por secuencias de pares $(palabras, (a, b))$. Las tuplas (a, b) son etiquetas complejas que pertenecen al conjunto Σ, donde a y b pertenecen a Σ_1 y Σ_2 respectivamente. Para inferir las etiquetas "simples" necesitamos *taggers* entrenados y para entrenarlos, material de entrenamiento. Creamos dos nuevas ocurrencias del material de entrenamiento, reemplazando cada etiqueta original (a, b), por la primera coordenada para una de las ocurrencias, y por la segunda coordenada para la otra. Las nuevas ocurrencias del material de entrenamiento son secuencias de tuplas de la forma $(palabra, a)$ y $(palabra, b)$. Descripto de otra manera, creamos dos nuevos conjuntos de secuencias de tuplas a partir de las secuencias de $(palabras, (a, b))$ remplazando los pares (a, b) por a para uno de los conjuntos, y por b para otro. Las funciones f_1 y f_2 son la proyección sobre la primera y segunda cordenada, respectivamente.

Con este nuevo material estamos en condiciones de entrenar por separado dos *POS Taggers* de tal forma que uno infiera etiquetas de Σ_1 y que el otro infiera etiquetas de Σ_2, como ilustramos en la Figura 2.1. Ahora para generar la etiqueta de Σ tenemos dos opciones:

- **Utilizar f.** Esto es, formar la tupla utilizando las etiquetas deducidas por los *taggers* entrenados con Σ_1 y Σ_2;

- **Cascada de *Taggers*.** Es decir, tratar las etiquetas de Σ_1 como palabras y los elementos de Σ_2 como las etiquetas a inferir. Para esto hay que generar el material compuesto por secuencias de tuplas (a, b). La idea es que a partir de una palabra w podemos inferir $a \in \Sigma_1$ y utilizando a inferimos $b \in \Sigma_2$. Combinamos a y b para obtener la etiqueta de Σ. Este método esta ilustrado en la Figura 2.2.

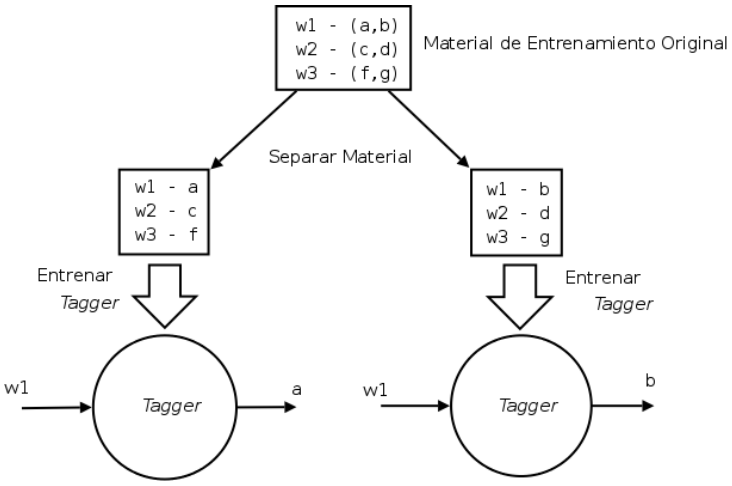

Figura 2.1: Generación de Material de Entrenamiento.

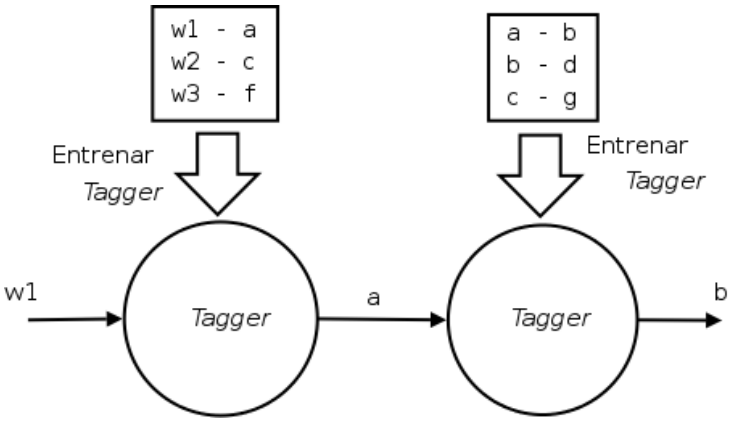

Figura 2.2: Generación de Material de Entrenamiento asoc. Σ_1 y Σ_2.

Un problema con estos métodos es la propagación de los errores: si los *taggers* cometen errores al inferir las etiquetas generamos una etiqueta errónea y no tenemos forma de detectarlo. Además existe la posibilidad de que generemos combinaciones de etiquetas que no existen en el conjunto Σ aunque algunas de estas combinaciones prohibidas son detectables a la hora de combinar las etiquetas. Discutimos los errores, con mayor profundidad, en el capítulo 5.

2.2. Implementación de las Funciones

Por lo general las funciones $f_1, \ldots, f_k$ son simples de implementar. Concretamente descomponen la cadena (secuencia de caracteres) que representa la etiqueta original en cadenas más chicas que denotan parte de la información que está representada en la etiqueta original. Las funciones de proyección dependen de la información representada en las etiquetas del conjunto Σ. Esto significa que no podemos dar una forma general de las funciones, y las decisiones de implementación se basan en una rigurosa observación del material de entrenamiento. En algunos casos es fácil determinar cómo dividir la etiqueta en partes "representativas". Un ejemplo simple son las etiquetas de *chunk* que se pueden separar en dos partes: una parte representa la posición de la palabra dentro del *chunk*, `B` que denota el inicio e `I` que denota el medio, y la parte restante que representa la clase de *chunk* al cual pertenece la palabra, `NP, VP, ADVP`, ..., etc. Además está la etiqueta que indica que esa palabra no está contenida por un *chunk*, esta etiqueta es `O`. En otros casos es muy difícil y recurrimos a heurísticas informales.

La función de recomposición f es más complicada porque depende de las decisiones que tomamos al implementar las funciones $f_1 \ldots f_k$. Además, no cualquier combinación de etiquetas de $\Sigma_1 \times \ldots \times \Sigma_k$ es necesariamente una etiqueta de Σ. Pueden existir combinaciones prohibidas, que no tienen sentido. Finalmente, si las etiquetas seleccionadas de los conjuntos $\Sigma_1, \ldots, \Sigma_k$ son incorrectas, el error se propaga a la composición de la etiqueta de Σ. La implementación de la "pila" de *taggers* es más simple. Concretamente es una cascada de *taggers* donde la salida de uno es la entrada del otro.

Capítulo 3

Taggers

3.1. Introducción

Tagging es la tarea de marcar (o etiquetar) cada palabra en una oración con su *part-of-speech* (o categoría morfosintáctica) apropiada. Decidimos si una palabra es nombre, verbo, adjetivo, o lo que corresponde. Aquí vemos un ejemplo de una oración etiquetada:

The-DT little-JJ orange-JJ ducks-NNS swallow-VBP flies-NNS.

Cada palabra puede pertenecer a distintas categorías léxicas, y en distintas oraciones cada palabra podría estar marcada con distintas etiquetas. En la siguiente tabla vemos listadas las etiquetas que admiten las palabras de la oración de arriba:

The	little	orange	ducks	swallow	flies
DT	JJ	JJ	NNS	VBP	NNS
		NN	VBP	NN	VBP

En este ejemplo vemos otra forma de etiquetar la oración:

The-DT little-JJ orange-NN ducks-VBP swallow-NN flies-VBP.

Aunque esta anotación conlleva a un lectura semánticamente incorrecta.

En *tagging* intentamos determinar cuál de todas las categorías léxicas es la más probable para un uso particular de una palabra en una oración. Dicho de otra manera: fijar las categorías morfosintácticas de cada palabra en una oración. A partir de material de entrenamiento, utilizando métodos de aprendizaje, generamos un modelo del lenguaje que nos permitirá clasificar

POS Tag	Significado	Ejemplo
DT	determiner	the
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
NN	noun, singular or mass	table
NNS	noun plural	tables
VB	verb, base form	take
VBP	verb, sing. present, non-3d	take

Cuadro 3.1: Algunas etiquetas y la categoría que representan.

las etiquetas de una oración. El esquema general está ilustrado en la Figura 3.1. Los métodos de aprendizaje o inferencia de modelos utilizan "contexto" para generar reglas, obtener estadísticas, o lo que el método de aprendizaje requiera para inferir un modelo. Describimos el contexto referido a tagging en la sección 3.2.

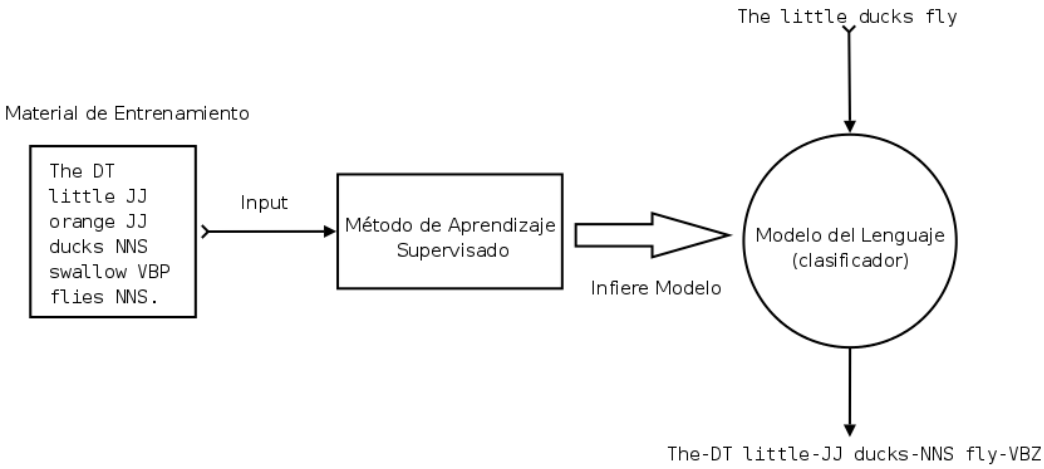

Figura 3.1: Esquema General

Los métodos más exitosos alcanzan entre un 96 % y un 97 % de precisión al desambiguar. De todas formas estos números no son tan buenos como

parecen ya que la evaluación es por palabra. Aún obteniendo una precisión del 96 %, en un texto cuyas oraciones alcanzan un promedio de 25 palabras, como son las publicaciones diarias, habrá en promedio más de un error *tagging* por oración.

Aunque es limitada, la información que obtenemos de *tagging* es muy útil. Extracción de la información, sistemas de *question-answering*, *shallow parsing* (chunking), son algunas de las aplicaciones que pueden aprovechar la información que se obtiene de *tagging*. *Tagging* funciona como un capa intermedia que permite generalizar la información, como así también mejorar la estadística.

En la sección 3.2 trataremos los *taggers* implementados con modelos de máxima entropía. Pero antes, como ya adelantamos, haremos unos comentarios generales sobre el tipo de contexto que se utiliza en *tagging*.

3.2. Fuentes de Información

¿Cómo se puede decidir la categoría correcta para una palabra dentro un contexto particular? Hay esencialmente dos fuentes de información. Una manera es observar las etiquetas de las otras palabras en el contexto de la palabra en la cual estamos interesados. Estas palabras también podrían ser ambiguas respecto a sus categorías, pero la observación esencial es que algunas secuencias de categorías como DT JJ NN (determinante - adjetivo - nombre) son comunes, mientras que otras son extremadamente improbables o imposibles, como DT JJ VBP (determinante - adjetivo - verbo), entonces a la hora de elegir entre las etiquetas NN (nombre, singular) o VBP (verbo, singular, presente) para anotar la palabra *play* en la frase *new play*, seleccionamos la primera etiqueta.

Conocer la palabra involucrada aporta mucha información sobre la etiqueta correcta. Por ejemplo, si bien en el Inglés *flour* puede ser usada como un verbo (*flour the dough*), es mucho más probable que *flour* ocurra como nombre. La utilidad de esta información es ilustrada por el hecho de que un *tagger* "tonto" que simplemente asigna la etiqueta más común a cada palabra trabaja a un nivel de un 90 % de exactitud [3].

Todos los *taggers* modernos usan de alguna manera una combinación de la información que provee la secuencia de etiquetas en el contexto de la palabra y la información léxica (la palabra que nos concierne).

La información léxica es muy útil ya que típicamente la distribución de los usos de una palabra a través de las distintas partes del discurso no es uniforme. Es decir, que las palabras dentro de un discurso tiendan a perte-

necer a la misma categoría dado que el uso (y por lo tanto su rol) tiende a ser el mismo a través del discurso. Aún para las palabras que pertenecen a un número de categorías, éstas usualmente son usadas como una categoría particular.

3.3. Máxima Entropía

Como ya mencionamos en el capítulo 1, una gran variedad de problemas en PLN pueden ser tratados como problemas de clasificación estadística, en los cuales la tarea es estimar la probabilidad de las ocurrencias de una clase lingüística a en un contexto b, o sea, $p(a, b)$. A partir de grandes corpus de textos, que nos proveen una fuente de información sobre la concurrencia de as y bs, nuestro problema es estimar, a partir de la evidencia, un modelo probabilístico $p(a, b)$.

Consideremos el Principio de Máxima Entropía que propone que la distribución correcta $p(a, b)$ es aquella que maximiza la entropía, o la "incerteza", conforme a restricciones que representan "evidencia" (i.e., hechos conocidos por el experimentador). En otras palabras, este principio supone modelar todo lo conocido, y no supone nada sobre lo que es desconocido. Esto es, dado un conjunto de restricciones que se deben cumplir (creemos que estas corresponden al "mundo real" que modelamos), seleccionamos un modelo que sea consistente con las restricciones y que en cualquier otro caso sea lo más "uniforme" posible.

Tomemos como ejemplo un dado "desconocido", el Principio de Máxima Entropía nos dice que deberíamos asumir una distribución uniforme. Ahora, supongamos que nuestro dado es sesgado y que le da una probabilidad de 0.4 y 0.2 al los números 4 y 6 respectivamente. La mejor distribución según el principio es:

1	2	3	4	5	6
0.1	0.1	0.1	0.4	0.1	0.2

Cualquier otra distribución haría suposiciones sobre la realidad.

Con mayor rigurosidad, si A denota el conjunto de las posibles clases y B denota el conjunto de posibles contextos, p, debería maximizar la entropía H :

$$H(p) = -\sum_{x \in \varepsilon} p(x) \log p(x)$$

donde $x = (a, b)$, $a \in A$, $b \in B$ y $\varepsilon = A \times B$, y debería mantenerse consistente con la evidencia, o "información parcial".

3.3.1. Representando Evidencia

Una manera de representar evidencia, es decir representar las restricciones del modelo, es codificar observaciones útiles como características (*features*) e imponer restricciones a los valores de las esperanzas de las características. Una característica es una función binaria sobre los eventos: $f_j : \varepsilon \rightarrow 0, 1$.

En PLN el término característica (*feature*) se refiere solamente a algún atributo de algún objeto, independientemente de la clase a la cual este objeto pertenece. En esta sección las características contienen información sobre la clase del objeto en cuestión, así también como "medidas" del objeto que queremos clasificar.

Dadas k características, las restricciones tienen las siguiente forma

$$E_p f_j = E_{\tilde{p}} f_j \ (1)$$

donde $1 \leq j \leq k$. $E_p f_j$ es la esperanza de f_j del modelo p:

$$E_p f_j = \sum_{x \in \varepsilon} p(x) f_j(x)$$

y está restringida a ser igual a la esperanza observada $E_{\tilde{p}} f_j$:

$$E_{\tilde{p}} f_j = \sum_{x \in \varepsilon} \tilde{p}(x) f_j(x)$$

donde $\tilde{p}$ es la probabilidad observada de x dentro de una muestra de entrenamiento S. Entonces, un modelo p es consistente con la evidencia observada si y solo si satisface las restricciones especificadas en (1). El Principio de Máxima Entropía recomienda que usemos p^*,

$$P = \{p \,|\, E_p f_j = E_{\tilde{p}} f_j, \, j = \{1 \ldots k\}\}$$

$$p^* = \mathbf{argmax}_{p \in P} \, H(p)$$

ya que maximiza la entropía sobre el conjunto de modelos consistentes P.

Se puede demostrar que p^* debe tener una forma equivalente a:

$$p^*(x) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x)}, \, 0 < \alpha_j < \infty$$

donde π es una constante de normalización y las α_js son los parámetros del modelo. Cada parámetro α_j corresponde a una característica (*feature*) f_j y se puede interpretar como el "peso" para esta característica. Los parámetros para la distribución $\alpha_1, \ldots, \alpha_k$ se pueden obtener utilizando el algoritmo *Generalized Iterative Scaling*. No veremos una demostración de esto ya que la demostración se encuentra en [11].

Características para *POS Tagging*

Dada una secuencia de palabras $w_1, \ldots, w_n$ y una de etiquetas $t_1, \ldots, t_n$ como material de entrenamiento, definimos h_i como la historia disponible al predecir t_i. En *POS Tagging* estimamos un modelo $p(h_i, t_i)$ y en este caso particular lo hacemos con el modelo probabilístico ya visto, esto es:

$$p(h,t) = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x)}$$

La probabilidad de una historia h y una etiqueta t está determinada por aquellos parámetros cuyas características correspondientes están activadas, i.e., aquellos α_j tal que $f_j(h,t) = 1$. Una característica, dado (h,t), podría activarse en cualquier palabra o etiqueta en la historia h, y debería codificar cualquier información que pudiera ayudar a predecir t, como la ortografía, o la identidad de las dos etiquetas anteriores. El contexto disponible para las funciones características está dado por la siguiente definición de h_i:

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

Por ejemplo,

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{si } \textit{sufijo}(w_i) = \text{“}\texttt{ing}\text{” y } t_i = \texttt{VBG} \\ 0 & \text{caso contrario.} \end{cases}$$

Si la característica existe en el conjunto de características del modelo, su correspondiente parámetro del modelo contribuirá a la probabilidad conjunta $p(h_i, t_i)$ cuando w_i termine en "*ing*" y t_i sea igual a `VBG`. Por tanto el parámetro del modelo α_j efectivamente sirve como un "peso" para ciertos "previsionistas contextuales", este caso el sufijo "`ing`", para la probabilidad de observar una cierta etiqueta, en esta caso `VBG`.

El modelo genera un espacio de características, explorando cada par (h_i, t_i) en el material de entrenamiento con una "plantilla" (*template*) dada en la Tabla 3.2. Dada h_i como la historia actual, una característica siempre hace una pregunta "si/no" sobre h_i, más aun restringe t_i a una cierta etiqueta. Las concretizaciones de las variables X,Y y T en la Tabla 3.2 son obtenidos automáticamente del material de entrenamiento. Para ver un tratamiento más profundo sobre el tema ver [10].

La generación de características (*features*) para etiquetar palabras desconocidas, aquellas que fueron vistas al entrenar, depende de la suposición de que las palabras "raras" en el material de entrenamiento, aquellas que aparecen menos de cinco veces en el material, son similares a la a las palabras

Condición	Característica	
	$w_i = X$	$\& \ t_i = T$
	$t_{i-1} = X$	$\& \ t_i = T$
	$t_{i-2}t_{i-1} = XY$	$\& \ t_i = T$
	$w_{i-1} = X$	$\& \ t_i = T$
	$w_{i-2} = X$	$\& \ t_i = T$
	$w_{i+1} = X$	$\& \ t_i = T$
	$w_{i+2} = X$	$\& \ t_i = T$

Cuadro 3.2: Características de la historia actual h_i

Word	The	little	orange	ducks	swallow	flies
Tag	DT	JJ	JJ	NNS	VB	NNS
Position	1	2	3	4	5	6

Cuadro 3.3: Datos

$w_i =$ `orange`	$\& \ t_i =$ `JJ`
$w_{i-1} =$ `little`	$\& \ t_i =$ `IN`
$w_{i-2} =$ `The`	$\& \ t_i =$ `IN`
$w_{i+1} =$ `ducks`	$\& \ t_i =$ `IN`
$w_{i+2} =$ `swallow`	$\& \ t_i =$ `IN`
$t_{i-1} =$ `JJ`	$\& \ t_i =$ `IN`
$t_{i-2}t_{i-1} =$ `DT JJ`	$\& \ t_i =$ `IN`

Cuadro 3.4: "*Features*" generadas a partir de h_3.

desconocidas en los "datos de testing[1]" con respecto a la forma en que se deletrean.

Por ejemplo la tabla del cuadro 3.3 contiene un extracto del material de entrenamiento mientras la Tabla 3.4 las características generadas al explorar (h_3, t_3) en donde la palabra actual es *orange*.

3.3.2. Probando el Modelo

El corpus de prueba se etiqueta de una oración a la vez. El procedimiento requiere una búsqueda para enumerar las secuencias de etiquetas candidatas para cada oración, y la secuencia de etiquetas con mayor probabilidad es seleccionada como respuesta.

El Algoritmo

El algoritmo de búsqueda, en esencia un *beam search*, usa la probabilidad condicional de una etiqueta

$$p(t|h) = \frac{p(h,t)}{\sum_{t' \in T} p(h,t')}$$

y mantiene, mientras consume una nueva palabra de la oración, las N secuencias candidatas con mayor probabilidad hasta ese punto en la oración. Dada la oración $w_1, \ldots, w_n$, una secuencia de etiquetas candidato $a_1, ..., a_n$ tiene la probabilidad condicional:

$$P(a_1, \ldots, a_n | w_1, \ldots, w_n) = \prod_{i=1}^{n} p(a_i | b_i)$$

donde b_i es la historia correspondiente a la palabra i.

Además el procedimiento de búsqueda opcionalmente consulta un *Diccionario de Etiquetas*, que, para cada palabra conocida, lista las etiquetas que han sido observadas, coocurriendo con la palabra, en el material de entrenamiento. Si se utiliza el diccionario, el procedimiento de búsqueda, para las palabras conocidas, generan etiquetas dadas por la entrada del diccionario, y para las palabras desconocidas genera todas las etiquetas en el conjunto. Sin el diccionario el procedimiento de búsqueda genera todas las etiquetas del conjunto. Para un tratamiento con más detalle ver [10, 11].

[1] Con "datos de testing" nos referimos al material de prueba que se utiliza para medir la eficacia del método.

3.4. Otros Métodos

Existen otros reconocidos métodos estadísticos para resolver *POS Tagging* como lo son aquellos que utilizan HMM (Modelo oculto de Markov). En el modelado del lenguaje con HMM los estados son las etiquetas y las emisiones son las palabras. A partir de una secuencia de palabras, utilizando Viterbi[2], recuperamos la secuencia de estados, y así inferimos la secuencia de etiquetas para la oración. Más información en [5].

Los modelos de Máxima Entropía ofrecen una manera clara de combinar diversas partes de evidencia contextual para estimar la probabilidad de cierta clase lingüística coocurriendo con cierto contexto lingüístico.

[2]El algoritmo de Viterbi permite encontrar las secuencia de estados más probable en un Modelo oculto de Markov

Capítulo 4

Chunking

4.1. Introducción

Chunking es un método robusto y eficiente para identificar frases cortas en un texto, o *chunks*. Los *chunks* son secuencias de palabras sintácticamente relacionadas que no se superponen, lo que significa que una palabra solo puede pertenecer a un *chunk*. Por ejemplo:

$[_{NP}$ He $][_{VP}$ reckons $][_{NP}$ the current account deficit $]$
$[_{VP}$ will narrow $][_{PP}$ to $][_{NP}$ only \$ 1.8 billon $]$
$[_{PP}$ in $][_{NP}$ September $]$

Chunking nos sirve para localizar información. En muchos sistemas de recuperación de información es útil tener indexadas todas las frases nominales para mejorar la búsqueda de expresiones terminológicas.

Además nos sirve para generalizar. Supongamos que deseamos estudiar patrones de algún verbo, por ejemplo `gave`. Utilizando chunking, podríamos generalizar las frases que le siguen al verbo suprimiendo las frase nominales que le siguen, por ejemplo, y así obtener patrones más abstractos.

```
gave NP
gave up NP in NP
gave NP up
gave NP to NP
```

4.2. Definiendo *Chunks*

Cuando leemos una oración, generalmente, la leemos de a pedazos o partes. Por ejemplo, la oración anterior la dividimos de la siguiente manera:

[Cuando leemos] [una oración], [la leemos] [de a pedazos].

Este ejemplo pertenece a Abney [2]. Su definición de chunk está inspirada en parte por estudios psicológicos que vinculan la duración de las pausas en la lectura y una estructuración simple de las oraciones a agrupaciones de texto que se llaman ϕ-phrases, que a grandes rasgos se corresponde a dividir la oración luego de cada núcleo sintáctico (*syntactic head*) que pertenezca a la categoría funcional (*content word*).

Más allá de esta justificación, dar una definición de qué es un *chunk* con exactitud es difícil. Las definiciones en la bibliografía difieren entre sí. Quizás la definición más simplista de un chunk es tomar todo lo que esté delimitado por palabras funcionales (o por su término en Inglés *stop words*) como *chunk*. Los que usamos en este trabajo son inferidos a partir del *Penn Treebank* [6, Marcus et al. 1994]. El método que se utiliza para inferir los *Chunks* se encuentra en [12]. En particular, empleamos el material de entrenamiento de la tarea de la *Conference on Computational Natural Language Learning* del año 2000 [CONLL 2000].

4.3. CONLL

CONLL es una conferencia anual sobre PLN. Todos los años presentan una tarea relacionada con un tema de PLN en donde los participantes desarrollan métodos para resolver dicha tarea. En el año 2000 la tarea fue producir métodos que puedan dividir un texto en frases: *chunking*.

Describiremos el material de entrenamiento de la competencia, como así también uno de los métodos que fueron presentados, además nombraremos otros métodos que pueden ser utilizados en *chunking*. En el Cuadro 4.1 vemos algunos resultados significativos de la competencia. El valor de referencia de la tarea está indicado en *baseline*. Osb00 indica los mejores valores obtenidos por Osborne [7], que explicamos a continuación, mientras que KM00 [4] fue el mejor resultado.

	Precisión	Recall	FB1
KM00	93.45 %	93.51 %	93.48
Osb00	91.65 %	92.23 %	91.94
baseline	72.58 %	82.14 %	77.07

Cuadro 4.1: Resultados significativos de CONLL 2000.

4.3.1. Material de Entrenamiento

El material de CONLL 2000 consiste en secuencias de oraciones divididas por saltos de línea. Cada palabra de las oraciones está anotada con una etiqueta de *chunk*. La etiqueta refleja el tipo del *chunk* al cual la palabra está asociada, así también si la palabra está en el medio del chunk (I - *in*) o al inicio (B - *begining*). Existen once "tipos" de *chunk*, cada uno denota alguna clase de sintagmas, o categorías sintácticas, estos son NP, VP, PP, ADVP, ADJP, CONJP, INTJ, LST, PRT, SBAR, O. El ultimo de los tipos (O) en realidad refleja que la palabra no esta en ningún *chunk* y en este trabajo nos referiremos a la notación BIO para describir la posición de una palabra en el chunk y no incluimos a O como un tipo. El material de entrenamiento también incluye el *POS Tag* de cada palabra. La siguiente tabla ilustra un extracto del material de entrenamiento.

Token	POS Tag	Chunk Tag
This	DT	B-NP
has	VBZ	B-VP
increased	VBN	I-VP
the	DT	B-NP
risk	NN	I-NP
of	IN	B-PP
the	DT	B-NP
government	NN	I-NP
.	.	O

4.3.2. *Chunking* tratado como *POS Tagging*

Según Osborne [7], tratar *chunking* como *POS Tagging* da resultados comparables a otros métodos más elaborados. El mejor modelo que propone tiene una "precisión al etiquetar" (*accuracy* por su término en Inglés) cada palabra de 94.88 % y un puntaje FB1 (definido a continuación) total de 91.44.

El método es bastante simple y consiste en entrenar un *POS Tagger* con diferentes *configuraciones* (concatenaciones) de palabras, *POS Tags*, *chunk types*, y/o sufijos o prefijos de palabras y/o *chunk types*. Utilizando estas concatenaciones en el entrenamiento, hacemos que el *tagger* utilice un modelo cuyas "palabras" codifican más información contextual que si lo hubiéramos entrenado solamente con palabras del lenguaje. Este contexto extra permite cerrar la brecha entre *POS Tagging* y *Shallow Parsing* (*Chunking*).

Se refiere a "configuraciones" a aquello a lo cual el *tagger* considera palabras. Una configuración será una concatenación de varios elementos del

material de entrenamiento relevantes al tomar una decisión a la hora de inferir un chunk. Un *POS Tag* podría ser parte de una configuración. Se refiere una predicción como lo que el *tagger* considera una etiqueta (*tag*). Las predicciones serán etiquetas correspondientes a los *chunks*.

Consideremos un fragmento del material de entrenamiento:

Palabra	w_1	w_2	w_3
POS Tag	t_1	t_2	t_3
Chunk	c_1	c_2	c_3

A partir del material de entrenamiento se construyen distintas configuraciones para predecir la etiqueta de chunk correspondiente a la palabra w_1. Generamos un material de entrenamiento para un *tagger* en donde "las palabras" son configuraciones y las etiquetas son los etiquetas de *chunking*.

Antes de continuar el tratamiento del método de Osborne, haremos un paréntesis para explicar las medidas de rendimiento utilizadas en *chunking*. Existen tres medidas, la primera es el porcentaje de frases detectadas correctamente (**precisión**). La segunda, el porcentaje de frases en los datos que fueron encontradas por el *chunker* (**recall**) - cuántas frases dentro del material de prueba fueron descubiertas por el *chunker*. Y la tercera es una combinación de ambas, la taza $F_{\beta=1}$ (o también **FB1**) está dada por (β^2+1)*precisión*recall / (β^2)*precisión+recall con $\beta = 1$ [13, van Rijsbergen, 1975]. Además nos referiremos a la "precisión al etiquetar" (*accuracy*) que es cuantas etiquetas fueron inferidas correctamente por el *chunker*.

Reproducimos dos de los resultados obtenidos por el método de Osborne. A partir de una configuración que consiste de palabra y etiqueta (w_1 y t_1), el método obtiene una precisión al etiquetar de un 93.71 % , una precisión de 89.62 % , recall de 90.61 % y el puntaje FB1 es de 90.11 %. El material de entrenamiento para este modelo está representado en la siguiente tabla:

configuración	$w_1 + t_1$	$w_2 + t_2$	$w_3 + t_3$
predicción	c_1	c_2	c_3

El mejor modelo que obtiene combina en un configuración la etiqueta actual t_1 la siguiente t_2, la etiqueta del *chunk* c_1, las últimas dos letras de la etiqueta del siguiente *chunk* c_2, las primeras dos letras de la palabra actual w_1 y las últimas cuatro letras de la palabra actual w_1. Los resultados son, precisión al etiquetar un 94.88 % , una precisión de 91.65 % , recall de 92.23 % y el puntaje FB1 es de 91.94 %.

Chunking se puede tratar de distintas maneras, como un problema de clasificación estadística, desde la teoría de lenguajes utilizando gramáticas libres de contexto, y expresiones regulares. Dirigimos al lector a las referencias [1, 12, 4] para conocer los diferentes métodos.

Lo que describe Osborne influyó bastante en las decisiones que tomamos en las experiencias que hicimos. Fue unos de los puntos de partida.

Capítulo 5

Experimentos

5.1. Introducción

Para validar nuestra hipótesis de trabajo desarrollamos numerosos experimentos de los cuales describimos solamente unos pocos. Previos experimentos nos permitieron generar intuición sobre los datos y en base a esta intuición y un poco de "prueba y error" tomamos las decisiones reflejadas en los experimentos que describimos. Algunas de las decisiones parecerán extrañas. Más adelante daremos nuestra explicación, pero por ahora nos concentraremos en los experimentos, los procedimientos y en los resultados.

En este capítulo vemos :

I. Distintas formas de descomponer las etiquetas complejas y recomponerlas. Esto es, distintas formas de definir $f_1, \ldots, f_k$ y f.

II. Evaluamos la inferencia de etiquetas simples, es decir la eficacia de los *taggers* entrenados con el material de etiquetas descompuestas. En otras palabras, luego de aplicar las funciones $f_1, \ldots, f_k$ al material de entrenamiento, entrenamos los *taggers* con el material resultante y medimos la precisión de los *taggers*.

III. Evaluamos los métodos de recomposición. Los compararemos respecto un *tagger* entrenado con el material original (el material que contiene las etiquetas complejas), esto es, evaluamos la precisión de la salida de la función f.

IV. Vemos cómo funcionan los métodos al reducir el tamaño del material de entrenamiento. Es decir, reducimos el material de entrenamiento y seguimos la misma secuencia de pasos (1, 2 y 3) descripta arriba.

Con esto deseamos observar que ocurre con nuestros métodos cuando entrenamos con material que tiene problemas de escasez de datos.

Describimos, en este capítulo, dos conjuntos de experimentos. Para el primer conjunto de experimentos utilizamos un material de entrenamiento artificial generado a partir del material de CONLL 2000 [12]. Decimos que el material es artificial porque las etiquetas que utilizamos no pertenecen a ninguna tarea de PLN en particular, éstas son concatenaciones de etiquetas de dos tareas conocidas que explicamos en los capítulos 3 y 4, *POS tagging* y *chunking* respectivamente. Las palabras en el material también son modificadas y más adelante veremos en detalle cómo fueron modificadas.

Para el segundo conjunto de experimentos generamos material de entrenamiento a partir del Cast3LB. El Cast3LB es un banco de árboles sintácticos que contiene pocos árboles sintácticos del Castellano, que entre otras cosas se puede utilizar para entrenar analizadores sintácticos [8]. Para ambos conjuntos de experimentos nuestro *baseline* (valor de referencia) siempre será la eficacia del *tagger* entrenado con el material original de cada experimento, es decir, la eficacia de un *tagger* al inferir etiquetas complejas.

Durante los experimentos utilizamos distintos formatos o configuraciones de etiquetas y palabras. Para no tener conflictos con los nombres llamaremos a lo que el *POS Tagger* considera como palabra una configuración y llamaremos predicciones a lo que el *POS Tagger* considera un etiqueta, tal como define Osborne [7]. Para etiquetar utilizamos un *POS Tagger* de Máxima Entropía implementado por Ratnahparki [10].

El capítulo sigue de la siguiente manera: primero, describimos el procedimiento general de los experimentos. Luego, desarrollamos los experimentos en los cuales utilizamos nuestro material artificial. Explicamos cómo generamos el material, definimos los métodos de descomposición y los de recomposición. Mostramos los resultados que obtuvimos con estos métodos y experimentamos reduciendo el material de entrenamiento. Por último desarrollamos los experimentos que hicimos utilizando el Cast3LB.

5.2. Procedimiento General

Describiremos en términos generales como hacemos cada experimento. Una vez que definimos las funciones de descomposición y recomposición los experimentos siguen de la siguiente manera:

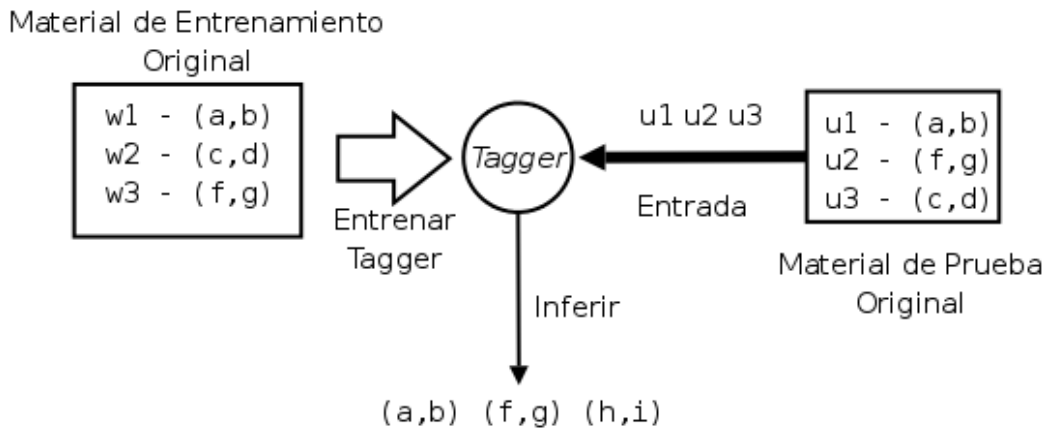

Figura 5.1: Primer Paso del Procedimiento General

I. Utilizando el material original sin modificar entrenamos un *tagger* para que infiera etiquetas complejas. Medimos la eficacia utilizando material de prueba. El material de prueba contiene oraciones etiquetadas no vistas durante el entrenamiento. Utilizamos estas oraciones para medir la eficacia del *tagger* dejando que éste infiera las etiquetas de las oraciones del material de prueba y luego comparamos las etiquetas inferidas contra las del material de prueba. Recordemos que éste será nuestro *baseline*, queremos ver si nuestros métodos pueden superar al *tagger* entrenado con etiquetas complejas.

II. Descomponemos el material original generando nuevo material de entrenamiento. Tantos materiales como funciones de proyección en la descomposición dado a que vamos entrenar a un *tagger* por cada conjunto de etiquetas simples generado (recordemos que es el material el que contiene las etiquetas).

III. Además generamos material de prueba para comprobar la eficacia de los *taggers* que infieren etiquetas simples. Para esto utilizamos las mismas descomposiciones del paso número 2 sobre el material de prueba original.

IV. Entrenamos los *taggers* para que infieran etiquetas simples utilizando el material que obtuvimos en el paso número 2.

V. Inferimos las etiquetas simples de cada material de prueba.

VI. Utilizando unos de los métodos de composición y la salida de cada *tagger* como entrada obtenemos el material de prueba etiquetado con

las etiquetas complejas inferidas por nuestro método de recombinación.

VII. Medimos la eficacia del método de la misma manera que lo hicimos con el método original. Comparamos las etiquetas obtenidas con las verdaderas, determinadas por un humano, que se encuentra en el material de prueba.

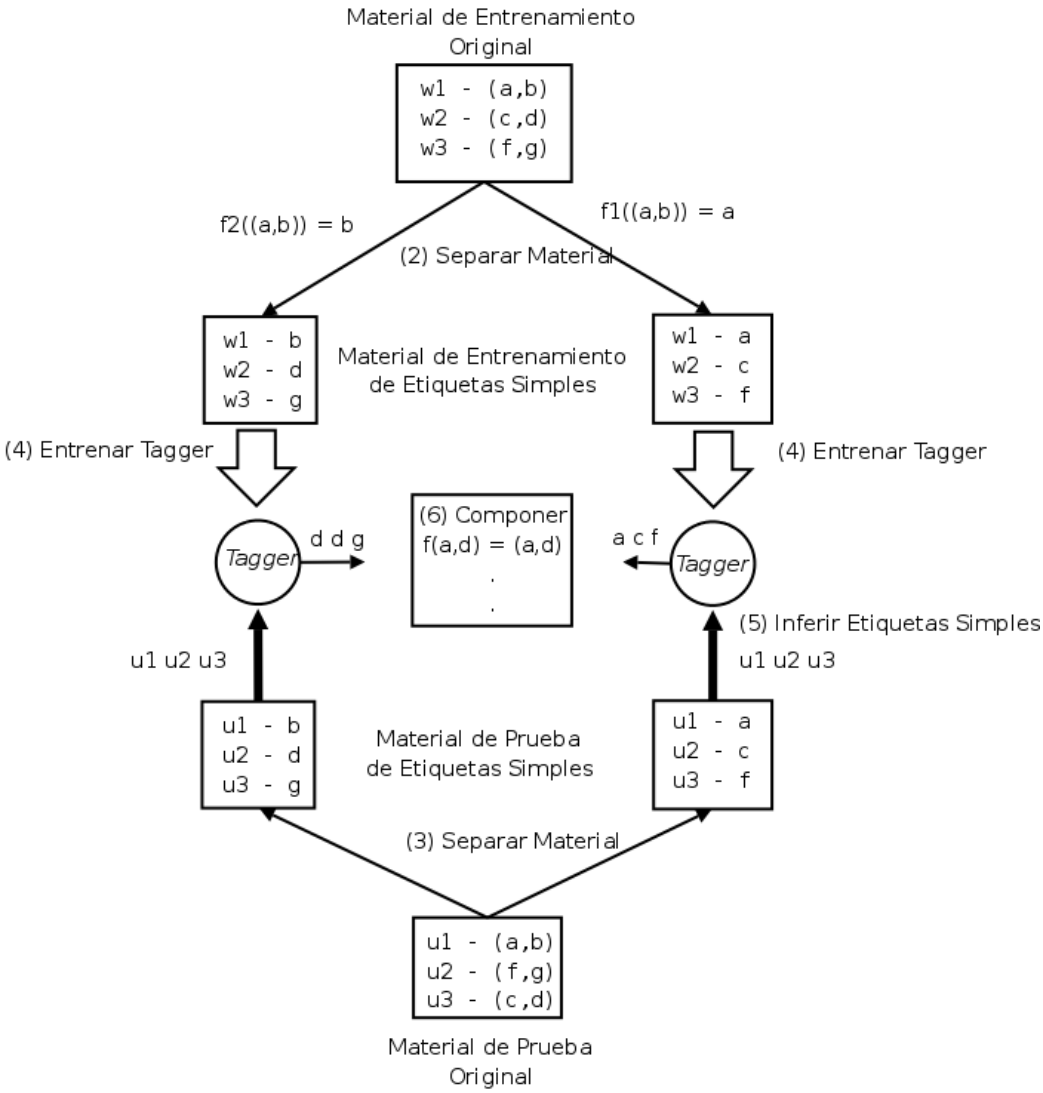

Figura 5.2: Procedimiento General

5.3. Experimentando con Material Artificial

Habíamos adelantado que experimentaríamos con material que generamos artificialmente. En esta sección explicamos cómo construimos el material a partir del material de CONLL 2000.

Primero daremos algunas definiciones. Como ya hemos visto en el capítulo 4, las etiquetas de *chunk* tienen dos partes: una que denota el tipo y otra que denota el inicio o el medio del *chunk*. Además contamos con una etiqueta que indica que la palabra marcada no está contenida en ningún *chunk*; recordemos que esta etiqueta es la O. Ahora definimos:

Chunktype como la parte de la etiqueta de *chunk* que representa al tipo del *chunk*. Además cuando hablamos de una ocurrencia de algún *chunktype* escribiremos `chunktype`.

BIO denotará la parte de la etiqueta de *chunk* que representa el inicio (`B`) o el medio (`I`) del *chunk* o ausencia de *chunk* (`O`). Para hablar de una ocurrencia en particular utilizaremos `bio`.

Token hará referencia a palabras o signos de puntuación. Denotamos una ocurrencia con `token`.

Abusamos de notación y utilizaremos estos términos para hablar de los conjuntos de estas entidades. Por ejemplo diremos que *chunktypes* es el conjunto de *chunktype*. Además escribimos `pos_tag` para referirnos a una ocurrencia de una etiqueta de *part-of-speech*. El acento circunflejo (ˆ) describe la concatenación entre las entidades lingüísticas anteriormente descriptas. Por ejemplo, `pos_tagˆchunktype`, denota la concatenación de una ocurrencia de un *POS tag* seguido de un *chunktype*. En la práctica utilizamos el circunflejo de divisor en la concatenación de las cadenas de caracteres que representan cada etiqueta.

En esta sección, utilizaremos el símbolo $\otimes$ para describir la composición de etiquetas para formar predicciones y configuraciones. Pretendemos utilizar este operador para describir conjuntos de predicciones y configuraciones a partir de símbolos más chicos. Podemos definir intuitivamente a $\otimes$ como un operador que toma dos conjuntos de etiquetas y como resultado de la operación devuelve un conjunto de etiquetas concatenadas. Con mayor rigurosidad, sean A y B dos conjuntos de etiquetas, entonces:

$$A \otimes B = \{\alpha\hat{}\beta \mid \forall \alpha \in A, \beta \in B\}$$

Token	*POS Tag*	*Chunks*
This	DT	B-NP
has	VBZ	B-VP
increased	VBN	I-VP
the	DT	B-NP
risk	NN	I-NP
of	IN	B-PP
the	DT	B-NP
government	NN	I-NP
.	.	O

Cuadro 5.1: Material de Entrenamiento de CONLL

¿Qué queremos decir con "definir intuitivamente"? Para empezar nunca trabajamos con el conjunto de etiquetas concretamente, sino que manipulamos el material de entrenamiento directamente y sería falso decir que existen todas las posibles combinaciones de etiquetas. Cuando deseamos obtener un material de entrenamiento que contenga ciertas predicciones, transformamos el material original sobre la marcha, tomando las etiquetas y separando las partes que necesitamos y volviendo a concatenar. Con esto nos gustaría dejar claro que las definiciones son más "reglas generales" que definiciones estrictamente hablando.

5.3.1. Generando Etiquetas Complejas

Partimos del material de CONLL 2000 para generar nuestro material inicial, esto es, el material que contiene al conjunto Σ de etiquetas. El material de entrenamiento de CONLL 2000 es un archivo de texto compuesto por tres columnas, en la primera se encuentran listados los *tokens*, en la segunda y en la tercer columnas se encuentran el *POS tag* y la etiqueta de *chunk* respectivamente. Una secuencia de filas que termina en una fila en blanco representa una oración o frase. Un extracto del material de entrenamiento de CONLL 2000 se encuentra en el Cuadro 5.1.

Generamos, a partir del material original, nuevo material concatenando las cadenas de caracteres que representan las etiquetas introduciendo algún símbolo que no se encuentra entre las etiquetas como divisor (como mencionamos anteriormente el acento circunflejo). Dicho de otra forma, las predicciones de este nuevo modelo serán elementos del conjunto *POS tags* $\otimes$ *Chunks*. Además, a los *tokens* le concatenamos su respectivo *POS tag*. Esto es, las configuraciones pertenecen al conjunto *Tokens* $\otimes$ *POS tags*. Está decisión es-

Configuración: *Tokens* $\otimes$ *POS tags*	Predicción: *POS tags* $\otimes$ *Chunks*
`This^DT`	`DT^B-NP`
`has^VBZ`	`VBZ^B-VP`
`increased^VBN`	`VBN^I-VP`
`the^DT`	`DT^B-NP`
`risk^NN`	`NN^I-NP`
`of^IN`	`IN^B-PP`
`the^DT`	`DT^B-NP`
`government^NN`	`NN^I-NP`
`.^.`	`.^O`

Cuadro 5.2: Material de Entrenamiento Artificial Resultante

ta guiada por los resultados de Osborne vistos en el capítulo 4. No es difícil observar que agregar el *POS tag* agrega regularidad y particulariza aún más el contexto. Ilustramos el material de entrenamiento resultante en el Cuadro 5.2.

Notemos que podríamos pensar que nuestro conjunto Σ son las etiquetas de *chunking* y que los *POS tags* son una característica (en el sentido de atributo) más con la que contamos para poder predecir el *chunk*.

Una vez que tenemos nuestro material, entrenamos el *tagger* y medimos la precisión al inferir las etiquetas compuestas (las predicciones). El material de CONLL 2000 cuenta con material de prueba al que le aplicamos la misma transformación y de esta forma contamos con material para medir la eficacia del *tagging*. Calculamos la precisión contando las predicciones correctas. Identificaremos a este modelo como "Modelo Original". Este procedimiento se corresponde con el primer paso de proceso descripto en la sección 5.2. La tabla siguiente resume las propiedades del modelo e indica la precisión que tiene.

Configuración	Predicción	Precisión
`token^pos_tag`	`pos_tag^chunk`	94.33 %

Como dijimos antes, podemos tratar a las predicciones como *chunks* con una característica adjunta, los *POS Tags*. Eliminando el *POS Tag* de la predicción y utilizando el programa `conlleval.pl` de CONLL evaluamos *chunking* a partir de las predicciones. Esto aporta a la evaluación general del método.

Evaluación de *Chunking*			
Accuracy	Precisión	Recall	FB1
94.39 %	90.58 %	92.25 %	91.4

5.3.2. Generando Etiquetas Simples

Generaremos dos conjuntos de funciones de proyección. Cada conjunto describe una manera de descomponer a un conjunto de etiquetas. El primer conjunto $\{f_1, f_2\}$ tiene como dominio a la composición de *POS tags* y *chunks*. El conjunto de llegada de f_1 se puede tratar como una composición de *POS tags* y *Chunktypes* $\cup$ $\{\epsilon\}$. Mientras que el conjunto de llegada de f_2 es una composición de *POS tags* y *BIO*.

$$f_1 : \textit{POS tags} \otimes \textit{Chunks} \rightarrow \textit{POS tags} \otimes \textit{Chunktypes} \cup \{\epsilon\}$$
$$f_2 : \textit{POS tags} \otimes \textit{Chunks} \rightarrow \textit{POS tags} \otimes \textit{BIO}$$

El segundo conjunto de funciones, $\{g_1, g_2\}$ tiene como dominio a la composición de *POS tags* y *Chunktypes* $\cup$ $\{\epsilon\}$ y como llegada ambos tienen lo que se puede tomar como una partición del conjunto, introduciendo algunos símbolos para discriminar ciertas etiquetas particulares.

$$g_1 : \textit{POS tags} \otimes \textit{Chunktypes} \cup \{\epsilon\} \rightarrow \textit{POS tags} \otimes \{\texttt{NP}, \texttt{VP}, \texttt{dummy}, \epsilon\}$$
$$g_2 : \textit{POS tags} \otimes \textit{Chunktypes} \cup \{\epsilon\} \rightarrow \textit{POS tags} \otimes \{\texttt{AVDP}, \texttt{PP}, \ldots, \texttt{dummy}, \epsilon\}$$

La descomposición que expresa el conjunto de funciones $\{f_1, f_2\}$ se puede tomar como la división de la etiqueta de *chunk* en *chunktype* y *BIO* agregando a cada parte de la etiqueta el *POS Tag*. Esto es, a partir de una predicción de la forma `pos_tag^chunk` dividimos la etiqueta en dos predicciones: `pos_tag^chunk` y `pos_tag^bio`. Hay que notar que si `bio` es igual a `O` (la etiqueta original es de la forma `pos_tag^O`, que significa que no hay tipo de `chunk`) entonces las predicciones resultantes son `pos_tag^`ϵ y `pos_tag^O`. En la tabla de abajo están tabulados los casos de cada función; en la primera columna están listados los posibles argumentos y en la segunda y tercera columna está el resultado de aplicar f_1 y f_2 respectivamente a dicho argumento. Observemos que podemos pensar que las f_i proyectan etiquetas de *chunks* manteniendo como característica contextual el *POS Tag* con el cual coocurren.

Predicción α	$f_1(\alpha)$	$f_2(\alpha)$
`pos_tag^B-chunktype`	`pos_tag^chunktype`	`pos_tag^B`
`pos_tag^I-chunktype`	`pos_tag^chunktype`	`pos_tag^I`
`pos_tag^O`	`pos_tag^`ϵ	`pos_tag^O`

Remplazando las etiquetas del material original por sus descomposiciones, creamos nuevo material de entrenamiento para entrenar un *tagger* que infiera las etiquetas simples formadas por *POS tags* y *chunktypes* y otro *tagger* para inferir las que son concatenaciones de *POS tags* y *BIO*. La siguiente tabla ilustra en forma conjunta el material de entrenamiento para cada *tagger*. En la primera columna vemos la configuración, mientras que en la segunda y tercera columna vemos las predicciones para cada *tagger*.

Configuración *Tokens* y *POS Tags*	Predicción (Σ_1) *POS Tags* y *Chunktypes*	Predicción (Σ_2) *POS Tags* y *BIO*
`This^DT`	`DT^NP`	`DT^B`
`has^VBZ`	`VBZ^VP`	`VBZ^B`
`increased^VBN`	`VBN^VP`	`VBN^I`
`the^DT`	`DT^NP`	`DT^B`
`risk^NN`	`NN^NP`	`NN^I`
`of^IN`	`IN^PP`	`IN^B`
`the^DT`	`DT^NP`	`DT^B`
`government^NN`	`NN^NP`	`NN^I`
`.^.`	`.`	`.^O`

A continuación describimos cómo funciona la descomposición que se genera a partir del conjunto de funciones $\{g_1, g_2\}$. Sea α una etiqueta que pertenezca a *POS tags*$\otimes$*Chunktypes*$\cup\{\epsilon\}$, si α es de la forma `pos_tag^chunktype` y si el *chunktype* es es `NP` o `VP`, entonces las etiquetas resultantes de la descomposición son $g_1(\alpha)$ = `pos_tag^chunktype` y $g_2(\alpha)$ = `pos_tag^dummy`[1]. En caso contrario, esto es, si en el material original el *token* estaba marcado con una etiqueta que tiene como *chunktype* a algo distinto de `NP` o `VP`, entonces las etiquetas resultantes son de la forma $g_1(\alpha)$ = `pos_tag^dummy` y $g_2(\alpha)$ = `pos_tag^chunktype`. Ahora si α es de la forma `pos_tag^`ϵ no la cambiamos, o sea, el resultado es `pos_tag^`ϵ para ambas funciones. Las funciones de proyección g_i que utilizamos para generar el material están definidas de la siguiente manera:

$$g_1(\texttt{pos_tag\^{}symbol}) = \begin{cases} \texttt{pos_tag\^{}chunktype} & \text{si } \texttt{symbol} = \texttt{chunktype} \text{ y es } \texttt{NP} \text{ o } \texttt{VP} \\ \texttt{pos_tag\^{}dummy} & \text{si } \texttt{symbol} = \texttt{chunktype} \text{ y no } \texttt{NP} \text{ ni } \texttt{VP} \\ \texttt{pos_tag\^{}}\epsilon & \text{si } \texttt{symbol} \neq \texttt{chunktype} \end{cases}$$

[1]El símbolo `dummy` lo utilizamos para indicar que ahí existía un tipo y no nos interesa saber cuál es. Necesitamos esta "muleta" para diferenciar una predicción con tipo de una predicción que tenga la forma `pos_tag^`ϵ, que indica ausencia de *chunktype* y por lo tanto ausencia de *chunk*.

g_2(pos_tag^symbol)

$$= \begin{cases} \texttt{pos_tag\^{}chunktype} & \text{si } \texttt{symbol} = \texttt{chunktype} \text{ y no es } \texttt{NP} \text{ ni } \texttt{VP} \\ \texttt{pos_tag\^{}dummy} & \text{si } \texttt{symbol} = \texttt{chunktype} \text{ y es } \texttt{NP} \text{ o } \texttt{VP} \\ \texttt{pos_tag\^{}}\epsilon & \text{si } \texttt{symbol} \neq \texttt{chunktype} \end{cases}$$

Siguiendo el mismo estilo de la tabla anterior, la siguiente tabla ilustra el material de entrenamiento para esta descomposición.

Tokens y POS Tags	POS y NP o VP	POS y AVDP o PP, etc
This^DT	DT^NP	DT^ϵ
has^VBZ	VBZ^VP	VBZ^ϵ
increased^VBN	VBN^VP	VBN^ϵ
the^DT	DT^NP	DT^ϵ
risk^NN	NN^NP	NN^ϵ
of^IN	IN^ϵ	IN^PP
the^DT	DT^NP	DT^ϵ
government^NN	NN^NP	NN^ϵ
.^.	.	.

5.3.3. Métodos de Combinación

Ahora pasamos a definir los métodos que utilizamos para combinar las etiquetas simples y formar la etiqueta compuesta. Definimos dos mecanismos directos, en contraste con la cascada de *taggers*, que veremos próximamente como método para etiquetar. Creamos dos métodos directos. Un método directo para la primera descomposición definida por las funciones $\{f_1, f_2\}$, para combinar etiquetas de las clases *POS tag* y *chunktype* y *POS tag* y *BIO* y recuperar las etiquetas del tipo *POS Tag* $\otimes$ *Chunk*. La otra heurística es para recuperar las etiquetas que pertenecen a *POS tag*$\otimes$ *Chunktype*$\cup\{\epsilon\}$ que particionamos de acuerdo al *chunktype*.

En términos generales los métodos directos recomponen las partes que fueron divididas en la descomposición concatenando los símbolos que fueron separados. Por ejemplo, dadas dos etiquetas simples `pos_tag^chunktype` y `pos_tag^bio` el resultado es `pos_tag^bio-chunktype`. Si las composiciones tienen errores detectables (generar etiquetas que no pertenecen a los *chunks*, por ejemplo cuando `bio` es igual a `O` y `chunktype` es igual a `NP`) el resultado de la composición es una etiqueta predefinida (por lo general la etiqueta vacía o fuera de chunk). Luego, veremos la implementación del método descripto como "cascada" de *taggers* que presentamos en la sección 2.2.

Métodos Directos

Hay dos métodos directos: uno que compone las funciones $\{f_1, f_2\}$ y otro método correspondiente a las funciones $\{g_1, g_2\}$. Estos métodos se corresponden con las funciones de composición f y g respectivamente.

$$f : (\textit{POS tags} \otimes \textit{Chunktypes}, \textit{POS tags} \otimes \textit{BIO}) \rightarrow \textit{POS tags} \otimes \textit{Chunks}$$

$$g : (\textit{POS tags} \otimes \{\texttt{NP},\texttt{VP},\texttt{dummy}, \epsilon\},$$
$$\textit{POS tags} \otimes \{\texttt{ADVP},\texttt{PP},\ldots,\texttt{dummy}, \epsilon\}) \rightarrow \textit{POS tags} \otimes \textit{Chunks}$$

El primer método, explicitado por la función f, toma como entrada dos etiquetas, una `pos_tag^chunktype` y otra etiqueta `pos_tag^bio` y si los *POS tags* son iguales concatenamos el símbolo `bio` con el símbolo `chunktype`. Ahora si el símbolo que representa la posición de la palabra en el *chunk* es `O`, entonces el *chunktype* tiene que ser ϵ (vacío). Si no lo es, hay un error, y por defecto tomamos que esa palabra está fuera del *chunk*, esto es, le asignamos como etiqueta de *chunk*, `O`. También puede ocurrir que el símbolo `chunktype` sea ϵ y el símbolo `bio` sea `B` o `I`, que identifican, "inicio" o "dentro" del *chunk*. Esto también es un error y por defecto la etiqueta resultante será `O`. La función f esta definida por la tabla siguiente.

Argumentos (Etiquetas Simples)		Composición
POS tags $\otimes$ *Chunktypes*	*POS tags* $\otimes$ *BIO*	Resultante
`pos_tag^chunktype`	`pos_tag^B`	`pos_tag^B-chunktype`
`pos_tag^chunktype`	`pos_tag^I`	`pos_tag^I-chunktype`
`pos_tag^chunktype`	`pos_tag^O`	`pos_tag^O`
`pos_tag^`ϵ	`pos_tag^O`	`pos_tag^O`
`pos_tag^`ϵ	`pos_tag^B`	`pos_tag^O`
`pos_tag^`ϵ	`pos_tag^I`	`pos_tag^O`

¿Qué ocurre si los *POS tags* son distintos? Combinamos las etiquetas de la misma manera, utilizando el *POS tag* del *tagger* con mayor precisión, que en este caso es el *tagger* que infiere la combinación de *POS tags* y *chunktypes*.

La siguiente tabla describe la función g que explicita el método de combinación para recuperar las etiquetas de la forma *POS tags* $\otimes$ *Chunktypes*.

Fila	Argumentos (Etiquetas Simples) $\{$NP,VP,dummy, $\epsilon\}$	$\{$ADVP,PP, . . . ,dummy, $\epsilon\}$	Composición Resultante
1	pos_tag^chunktype	pos_tag^dummy	pos_tag^chunktype
2	pos_tag^chunktype$_1$	pos_tag^chunktype$_2$	pos_tag^chunktype$_1$
3	pos_tag^chunktype	pos_tag^ϵ	pos_tag^ϵ
4	pos_tag^dummy	pos_tag^chunktype	pos_tag^chunktype
5	pos_tag^dummy	pos_tag^dummy	pos_tag^ϵ
6	pos_tag^dummy	pos_tag^ϵ	pos_tag^ϵ
7	pos_tag^ϵ	pos_tag^ϵ	pos_tag^ϵ
8	pos_tag^ϵ	pos_tag^dummy	pos_tag^ϵ
9	pos_tag^ϵ	pos_tag^chunktype	pos_tag^ϵ

En este método el *tagger* más eficaz es el infiere predicciones que tienen *chunktype* igual a NP o VP. Es por esto que en la fila 3 vemos que en la composición resultante el *chunktype* pertenece al primer argumento de la función. Las filas 1,4,7 son los casos que no hay error. Es decir, son los casos en que los *taggers* coinciden.

Cascada de *Taggers*

Para este método necesitamos generar un nuevo material de entrenamiento que asocia las etiquetas pos_tag^chunktype y pos_tag^bio, y de esta manera inferir, utilizando como configuraciones a las etiquetas pos_tag^chunktype, las predicciones de la forma pos_tag^bio. A este método lo podríamos definir de manera similar a la función f. Se diferencian en cómo lo implementamos. En este caso, en vez de tomar dos etiquetas y concatenarlas, tomamos una e inferimos la segunda etiqueta y luego combinamos ambos símbolos. La combinación consiste simplemente en regenerar la etiqueta *chunk* utilizando la configuración y la predicción, sin eliminar el *POS tag*. Se combina de la misma manera que el primer método directo.

5.3.4. Resultados

Una vez que generamos el material de entrenamiento (de etiquetas simples) para cada *tagger* lo duplicamos y entrenamos con el material duplicado. Esto es porque el algoritmo de máxima entropía descarta características que ocurran menos de 10 veces y dado que el material es chico, perdemos mucho contexto. El Cuadro 5.3 muestra la eficacia de los *taggers* entrenados con etiquetas simples.

Una vez entrenados los *taggers* para inferir etiquetas simples (los *taggers* son: B, C, D, E ,F) aplicamos los métodos de combinación de la siguiente

Tagger	Configuración	Predicción	Precisión
B	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ *Chunktypes*	96.27 %
C	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ *BIO*	96.01 %
D	*POS tags* $\otimes$ *Chunktypes*	*POS tags* $\otimes$ *BIO*	98.78 %
E	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ $\{$`NP`,`VP`,`dummy`,$\epsilon\}$	97.05 %
F	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ $\{$`ADVP`,`PP`,...,`dummy`,$\epsilon\}$	96.77 %
Comp. Directa de E y F[a]		*POS tags* $\otimes$ *Chunktypes*	96.14 %

[a]Composición de los *taggers* mediante el método directo de la función g.

Cuadro 5.3: Evaluación de los *taggers* al inferir etiquetas simples

manera: combinamos la salida de los *taggers* B y C utilizando la función f; combinamos los *taggers* B y D utilizando el método de cascada; combinamos los *taggers* E y F con la función g y así recuperamos *POS tags*$\otimes$*Chunktypes* y componemos la salida con el *tagger* D. En las siguientes tablas mostramos la evaluación de la eficacia de los métodos de composición de etiquetas simples. En el Cuadro 5.4 vemos resumidos los resultados de la evaluación de métodos para recuperar etiquetas complejas y en la tabla que sigue, Cuadro 5.5, los resultados de la evaluación de *chunking*.

Modelos que Combinamos		Método	Predicción	Precisión
Modelo 1	Modelo 2	de Comp.	Resultante	
Modelo Original		-	*POS tags* $\otimes$ *Chunks*	94.33 %
B	C	Directo	*POS tags* $\otimes$ *Chunks*	94.06 %
B	D	Cascada	*POS tags* $\otimes$ *Chunks*	94.31 %
Comp. E y F	D	Cascada	*POS tags* $\otimes$ *Chunks*	94.17 %

Cuadro 5.4: Evaluación al Inferir Predicciones

Modelos que Combinamos		Chunking Evaluación			
Modelo 1	Modelo 2	Acc.	Precisión	Recall	FB1
Modelo Original		94.39 %	90.58 %	92.25 %	91.4
B	C	94.10 %	89.60 %	92.08 %	90.82
B	D	94.35 %	90.37 %	91.71 %	91.03
Comp. E y F	D	94.22 %	90.67 %	91.57 %	91.12

Cuadro 5.5: Evaluación de *Chunking*

Como podemos deducir de las tablas los métodos de composición no logran superar al *tagger* entrenado con el material que contiene las etiquetas complejas. Nos preguntamos si esto es causa de los métodos de composición rústicos y simplistas, o si la inferencia de las etiquetas simples no es lo suficientemente buena. Para resolver este planteo proponemos descomponer la salida del "Modelo Original" y evaluaremos la eficacia del modelo original al inferir las etiquetas simples. Simplemente aplicamos las funciones f_1, f_2 a las etiquetas complejas inferidas por el "Modelo Original" y evaluamos utilizando el material de prueba de los modelos B y C. Como se puede observar en el Cuadro 5.6, y comparando contra el Cuadro 5.3, el método original también es más eficaz al inferir etiquetas simples.

Tipo de Etiqueta Inferida	Precisión	Diferencia respecto los *taggers* de etiquetas simples
POS tags $\otimes$ *Chunktypes*	96.36 %	0.09 %
POS tags $\otimes$ *BIO*	96.04 %	0.03 %

Cuadro 5.6: Precisión de la Inferencia de Etiquetas Simples Utilizando el "Modelo Original"

5.3.5. Reduciendo la Cantidad de Material

Nuestros métodos apenas igualan al método convencional al inferir etiquetas complejas. ¿En qué contexto podrían nuestros métodos superar a los métodos convencionales? Deseamos probar lo que ocurre con materiales de entrenamiento que tienen problemas de escasez de datos. Esto se logra fácilmente reduciendo el material. Decidimos entrenar un método de combinación y el método convencional con menos material de entrenamiento.

I. Tomamos la primera mitad y el primer cuarto del material de CONLL 2000.

II. Generamos material artificial, el que contiene las etiquetas complejas, a partir de esta mitad y este cuarto. Es decir ahora tenemos medio material artificial y un cuarto de material.

III. Entrenamos dos *taggers* uno con la mitad del material artificial y otro con el cuarto.

IV. Con el material artificial reducido, utilizamos los mismos métodos de descomposición generando los materiales de entrenamiento necesarios,

en este caso, para inferir etiquetas de la clase *POS tags* $\otimes$ *Chunktypes* y el material que asocia que tiene como configuraciones a etiquetas de *POS tags*$\otimes$*Chunktypes* y predicciones que pertenecen a *POS tags*$\otimes$*BIO*.

v. Entrenamos los *taggers* necesarios para aplicar los métodos de recombinación con este material. Seguimos el mismo procedimiento de duplicar el material antes de entrenar.

Los resultados parciales, esto es, la eficacia al inferir predicciones simples, están listados en las tablas 5.7, 5.8, 5.9. La cantidad material de prueba no varía, es el mismo que el de los experimentos anteriores.

Tagger	Configuración	Predicción
Original	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ *Chunks*
B	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ *Chunktypes*
D	*POS tags* $\otimes$ *Chunktypes*	*POS tags* $\otimes$ *BIO*
Comp. B y D	*Tokens* $\otimes$ *POS tags*	*POS tags* $\otimes$ *Chunks*

Cuadro 5.7: Configuraciones y Predicciones

Modelo	Precisión con $\frac{1}{2}$ Material	Precisión con $\frac{1}{4}$ Material
Original	93.57 %	92.70 %
Comp. B y D	93.91 %	93.22 %
B	95.99 %	95.46 %
D	98.57 %	98.35 %

Cuadro 5.8: Precisión de los *taggers*

En general al disminuir el material de entrenamiento, los métodos propuestos empeoran la eficacia, disminuye la precisión de todos los *taggers*, aunque también, podemos observar que el método de composición que utilizamos es más eficaz que el método original.

En el Cuadro 5.10 vemos los resultados de repetir el experimento para evaluar al "Modelo Original" en la inferencia de etiquetas simples utilizando menos cantidad de material de entrenamiento. Hicimos el experimento para la clase de etiquetas *POS tags* $\otimes$ *Chunktypes* solamente, ya que el método de composición que utilizamos (cascada) utiliza como configuración a éstos. Generar una descomposición de la salida de "Modelo Original" para evaluar la composición *POS tags* $\otimes$ *BIO* no sería apropiado ya que estaríamos comparando distinto tipo de información y distinto tipo de modelo.

Cantidad de Material	Modelo	*Chunking* - Modelo Original			
		Accuracy[c]	Precisión	Recall	FB1
$\frac{1}{2}$	Original	93.64 %	89.06 %	91.36 %	90.16
	Comp. B y D	93.97 %	89.40 %	91.19 %	90.26
$\frac{1}{4}$	Original	92.81 %	87.55 %	90.23 %	88.87
	Comp. B y D	93.31 %	88.14 %	90.42 %	89.27

[c]Precisión de la asignación de las etiquetas de *chunking*. Utilizamos esta terminologia para diferenciar la Precisión del *chunking*

Cuadro 5.9: Evaluación de *Chunking*

Etiqueta Simple Inferida	Precisión	Diferencia respecto los *taggers* de etiquetas simples
Medio Material	95.99 %	0 %
Un Cuarto de Material	95.49 %	0.03 %

Cuadro 5.10: Evaluación de la descomposición de la salida del "Modelo Original", solamente para las etiquetas del conjunto *POS tags* $\otimes$ *Chunktypes*

5.4. Cast3LB

El Cast3LB es un banco de árboles sintáctico del castellano. Lo que nos interesa a nosotros son las etiquetas para representar la información morfológicas de las palabras. Este conjunto de etiquetas se basa en las etiquetas propuestas por el grupo EAGLES para la anotación morfosintáctica de lexicones y corpus para todas las lenguas europeas. Las etiquetas EAGLES compilan mucha información morfosintáctica sobre las palabras. La información se resume en una etiqueta compuesta de segmentos donde cada segmento es algún atributo que tiene la categoría que representa la etiqueta. Hay 12 categorías, entre ellas adjetivos, adverbios, determinantes, nombres, verbos, pronombres, etc. Cada categoría tiene un formato distinto para sus etiquetas. El formato expresa qué atributo es cada segmento. Lo interesante de este conjunto de etiquetas complejas es que se pueden descomponer según sus atributos. Y en efecto utilizamos esta propiedad para descomponer las etiquetas en dos conjuntos.

En el Cuadro 5.3 vemos una etiqueta EAGLES que representa la conjugación del verbo cantar en el futuro de subjuntivo, es decir, cantáremos. Como dijimos, cada segmento representa un atributo de la categoría. En la posición 1, está la categoría de la etiqueta, en la segunda posición el tipo (principal, auxiliar, semiauxiliar), en la tercera el modo, etc. Los segmentos

que nos interesan para esta aplicación son el sexto y el séptimo, número y género respectivamente.

V	M	S	F	1	P	0
1	2	3	4	5	6	7

Figura 5.3: Una ocurrencia de una etiqueta EAGLES

Nuestro método de descomposición se basa en las etiquetas de género y número. A partir de una etiqueta, generamos dos nuevas etiquetas que son una copia de la original solamente que eliminamos de una de las etiquetas el género y de la otra el número, se podría decir que particionamos en genero y numero.

Etiqueta Sin Género:

V	M	S	F	1	P
1	2	3	4	5	6

Etiqueta Sin Número:

V	M	S	F	1	0
1	2	3	4	5	6

Figura 5.4: Descomposición de las etiquetas

El experimento continúa como sigue:

I. A partir del material del Cast3LB extraemos las oraciones con sus respectivas etiquetas representantes de la categoría morfosintáctica. El Cast3LB cuenta con dos conjuntos de árboles; uno de entrenamiento y otro de prueba. Extraemos las oraciones de ambos y de esta forma también generamos material de prueba.

II. Con este nuevo material entrenamos un *tagger* para que infiera las etiquetas de morfología. Esto es análogo al *tagger* que infiere las etiquetas complejas en el experimento anterior. Utilizamos el material de prueba para medir la eficacia de este *tagger*. Esto será nuestro *baseline* para este experimento.

III. Dividimos las etiquetas que representan las siguientes categorías: adjetivos, adverbios, determinantes, nombres, verbos, pronombres en dos.

Para una de los descomposiciones eliminamos el atributo de género y en la otra eliminamos el atributo de número. Esto es, se generalizan las etiquetas eliminando número o género. Hacemos lo mismo para el material de prueba.

IV. Generamos dos nuevos materiales de entrenamiento. Uno donde las etiquetas de las categorías seleccionadas no tienen número y otro donde no tienen género. Esta descomposición es simple de implementar.

V. Entrenamos dos *taggers*, uno por material generado en el punto anterior. Esto es, entrenamos un *tagger* con un y sólo un material generado en el punto anterior.

VI. Medimos la eficacia de los *taggers* que infieren las descomposiciones de las etiquetas, esto se pueden ver en el Cuadro 5.11. Este punto también generamos la entrada para los métodos de composición.

VII. Utilizamos el método de composición que explicaremos a continuación para generar las etiquetas completas de morfología y medimos la eficacia de la composición. En el Cuadro 5.12 vemos los resultados de los métodos de composición.

Tagger	Precisión
A - Eliminado Genero	93.218 %
B - Eliminado Número	93.030 %

Cuadro 5.11: Precisión al Inferir Etiquetas Simples

Método	Precisión
Original	92.733 %
Composición de A y B	91.924 %

Cuadro 5.12: Precisión al Inferir Etiquetas Complejas

Método de Composición

Aquí desarrollamos el método de composición que utilizamos para componer las etiquetas. Dadas dos etiquetas, procedemos de la siguiente manera:

I. Tomamos las dos etiquetas y verificamos si ambas etiquetas representan la misma categoría inspeccionando el primer segmento de ambas etiquetas.

II. Si **no son iguales**, devolvemos la etiqueta del *tagger* con mayor eficacia. En el caso que la etiqueta pertenezca a las categorías que descomponemos, adjetivos, adverbios, determinantes, nombres, verbos, pronombres, completamos la etiqueta con género "masculino" para todas las categorías menos los verbos que los completamos con "0" (que significa que no está especificado).

III. Devolvemos la etiqueta del *tagger* con mayor eficacia aún cuando sea un error (cuando las categorías son iguales, pero las etiquetas son distintas, por ejemplo, nombre propio y nombre común).

IV. Si la categorías **son iguales** y la categoría de la etiqueta **es** adjetivos, adverbios, determinantes, nombres, verbos, pronombres, entonces, primero, hay que verificar si las etiquetas son iguales en todos los atributos sin tener en cuenta los atributos eliminados, género y número. Con esto detectamos el error que la categoría es la misma, pero los objetos dentro de la categoría (y por lo tanto sus etiquetas) son distintos. Por ejemplo, supongamos que la etiqueta inferida por ambos *taggers* fue un nombre, pero uno de los *taggers* infirió un nombre común y el otro un nombre propio, son etiquetas distintas y el género que infirió un *tagger* no corresponde con el *número* que infirió el otro. Respecto de estos errores, no los corregimos, los contamos para posterior estadística que presentaremos en las conclusiones.

V. Sin tener en cuenta si ocurrió el error descripto arriba, combinamos las etiquetas insertando el segmento del atributo número de la etiqueta inferida por el *tagger* entrenado con el material que generaliza el género (mantiene el número) en la otra etiqueta inferida por el *tagger* entrenado con el material que generaliza el número. Resumiendo, le agregamos el atributo número a la etiqueta inferida que no lo tiene (la etiqueta que tiene el género solamente). La salida es la etiqueta modificada.

Los resultados son negativos, el método de composición no logra superar el valor de referencia y en este caso es muy notable la diferencia. Supusimos que el problema era el mismo que el de los experimentos anteriores, es decir, que la inferencia de las etiquetas simples era mala. Para nuestra sorpresa no es así. Utilizamos la misma descomposición en la salida del "Modelo Original" de este experimento y medimos la eficacia del "Modelo Original" al

inferir etiquetas simples. Es decir, extraemos etiquetas simples de las etiquetas complejas inferidas por el "Modelo Original" y comparamos el resultado contra el material de prueba de los modelos que infieren etiquetas simples. En la siguiente tabla (Cuadro 5.13) vemos los resultados.

Tagger	Precisión	Diferencia respecto los *taggers* de etiquetas simples
Eliminado Genero	93.16 %	- 0.06 %
Eliminado Número	92.79 %	- 0.24 %

Cuadro 5.13: Precisión al Descomponer el resultado del "Modelo Original"

Al inspeccionar los resultados de cada *tagger* notamos que los conjuntos de errores que cometen los "Modelos Simples" no están contenido en el conjunto de errores que comete el "Modelo Original", es decir, no cometen los mismos errores. Analizamos los resultados de inferir etiquetas simples. Tomamos los resultados de los experimentos con las etiquetas sin el atributo género. El material de prueba cuenta con 7417 etiquetas, de las cuales 6819 etiquetas están bien inferidas por ambos modelos, es decir, el 91.93 % de las etiquetas del material de prueba, las inferidas por el "Modelo Original" y por el *tagger* entrenando con etiquetas simples directamente son iguales entre sí. El "Modelo Original" cometió 507 errores de los cuales 95 de esas etiquetas fueron bien inferidas por el *tagger* entrenado con etiquetas simples, mientras que éste cometió 503 errores de los cuales 91 no son errores del "Modelo Original". Inspeccionando la salida no observamos ningún patrón de error significante. Un fenómeno similar ocurre con las etiquetas sin el atributo número. La cantidad de etiquetas es la misma. La cantidad de etiquetas bien inferidas por ambos modelos es de 6810, esto es, el 91.80 %. En este caso el "Modelo Original" cometió 535 errores de los cuales 90 no fueron hechos por el *tagger* entrenado con etiquetas simples. Análogamente, el *tagger* hizo 517 errores de los cuales 72 no son errores del "Modelo Original". El 96 % de las etiquetas que se combinan son iguales o son etiquetas descompuestas, cuyos atributos que no son, ni género ni número, coinciden (ver el paso 4 del método de composición del Cast3LB). Más allá de todos estos datos, no podemos explicar lo que sucede al descomponer la salida del "Modelo Original" con los experimentos que presentamos.

Capítulo 6

Conclusiones

En el trabajo realizado se presentamos una posible solución al problema de escasez de datos en el material de entrenamiento, focalizándonos en *POS Tagging* y etiquetas complejas. Trabajamos con etiquetas que se puedan descomponer en partes significativas que reemplazan a las etiquetas en el material de entrenamiento. Nuestra hipótesis es que reemplazar las etiquetas por partes significativas genera mayor regularidad en los datos, es decir, en el material de entrenamiento encontraremos una mayor cantidad de patrones regulares de partes de etiquetas (o etiquetas simples). En el capítulo introductorio nos hicimos las siguientes tres preguntas:

I. ¿Qué formas de descomposición existen?

II. ¿Qué ventajas tendría nuestro método con respecto al método original?

III. ¿Los métodos de recomposición podrían agregar errores?

En esta sección respondemos a cada una de éstas:

Como explicamos en el capítulo 5, las formas de descomponer etiquetas dependen de la información que representan. Dado un conjunto de etiquetas, buscamos reconocer subestructuras significativas dentro de las etiquetas. Luego de que hayan sido definidas las subestructuras, utilizamos las definiciones para descomponer las etiquetas originales. En algunos casos, las mismas subestructuras identificadas pueden ser etiquetas simples aunque en todos los experimentos que hicimos concatenamos varias de ellas para formar las etiquetas simples. Por ejemplo, en el caso de los *chunks* sabemos que una etiqueta, que no sea `O`, tiene dos partes, `B` o `I`, seguido de algún tipo de *chunk*. La manera intuitiva de dividir las etiquetas es en estas dos partes, las que identifican, inicio o medio, y el tipo. Sin embargo, en otros casos la

descomposición no es trivial. En el caso de las etiquetas EAGLES eliminamos segmentos para simplificar (generalizar información) las etiquetas. ¿Por qué no dividimos la etiqueta en cualquier parte? ¿Por qué no tener una descomposición que divida la etiqueta a la mitad según su longitud de caracteres? Si descomponemos las etiquetas de una manera desprolija, sin fundamentos, perdemos los patrones que las secuencias de las etiquetas generan. El resultado de una descomposición "no pensada" puede resultar en la pérdida de los patrones y la obtención de "ruido". A la hora de descomponer las etiquetas siempre dejamos información que aporta a formar un patrón regular y separar información más especifica en las etiquetas simples. Por ejemplo, en la introducción generalizábamos los verbos y formamos otro conjunto de etiquetas con la letra que diferencia la conjugación; y en el experimento de Cast3LB separamos los atributos de género y número. Cuando trabajamos con palabras en las configuraciones, proponemos como regla general, siempre dejar información sobre el rol de la palabra, no necesariamente los *POS Tags*, pero si algo de la información que aportan. Cuando trabajamos con el Cast3LB, las etiquetas tienen el tipo y la categoría. Esto se debe a que los patrones de *POS Tags* aportan mucha información regular. Buscar la mejor descomposición es un balance entre generalidad y especificidad de las etiquetas.

Los métodos de composición son rústicos y simples, y éstos no llegan a alcanzar la eficiencia del método regular. En todos los experimentos vimos que el *tagger* entrenado con las etiquetas complejas infiere con mayor eficacia que los métodos de composición. Si bien las diferencias entre el "Modelo Original" y el método de recomposición no son muy significativas, en la sección 5.3.5, observamos que al reducir el tamaño del material de entrenamiento uno de los métodos, la cascada de *taggers*, mejora su eficacia con respecto al "Modelo Original".

Además, en los experimentos descriptos en la sección 5.3.4, observamos que los *taggers* entrenados con etiquetas simples no superan la precisión medida al descomponer la salida del "Modelo Original", aunque la diferencia de precisión entre los modelos es no mayor al 0.09 %. Al contrario con lo que ocurre con los experimentos hecho con material artificial, en los experimentos hechos con el Cast3LB los *taggers* entrenados con etiquetas simples superan la precisión medida al descomponer la salida del *tagger* entrenado con las etiquetas complejas. Aunque en este caso, la diferencia de precisión tampoco es de una magnitud significativa, la diferencia más grande que hay es del 0.27 %. Es probable que este fenómeno, se deba a diferencia que existe entre las estructuras de las etiquetas artificiales y las EAGLES. La transformación que le hacemos a las etiquetas EAGLES, extraidas del Cast3LB,

es una generalización, es decir, ampliamos su campo de representación al disminuir los rasgos propios de las etiquetas, se "regularizan", y de esta manera se generan una mayor cantidad de patrones regulares en el material de entrenamiento. Por ejemplo, cuando eliminamos la anotación que indica la conjugación verbal de las etiquetas que representan los verbos en el ejemplo de la introducción. En cambio, a las etiquetas artificiales las descomponemos a partir de dos características e inferimos estas características por separado. Aunque quizás exista una relación entre ambas características que aporta información que mejora la inferencia mutua entre ambas características y cada característica por separado. Por ejemplo, la probabilidad de que luego de `B-NP` siga `I-NP` o `B-NP` es difícil de capturar al separar las etiquetas. Los *taggers* en "cascada" intentan capturar esta relación aunque queda por evaluar esta metodología. Suponemos que nuestros métodos de composición funcionarán eficazmente en un ámbito donde las etiquetas estén compuestas de subestructuras independientes entre sí.

Respecto la implementación de los métodos, vimos que no sé puede crear una forma general para descomponer las etiquetas ya que depende de la información que las mismas representan. Esto implica que hay que implementar distintos métodos de descomposición para cada conjunto de etiqueta con el cual vayamos a trabajar, y luego entrenar tantos *taggers* como etiquetas simples definidas.

Al combinar etiquetas podemos detectar si la combinación resultante tiene sentido aunque no podamos a saber si la etiqueta es la correcta. En el caso del material artificial, podemos hacer esto porque conocemos la estructura de las etiquetas complejas. Sabemos que los símbolos, `B` e `I`, son seguidos por un tipo de *Chunk* y sabemos que cualquier combinación con el símbolo `O` está prohibida. En el caso de las etiquetas EAGLES, podemos detectar los errores verificando que los segmentos no eliminados de ambas etiquetas sean el mismo. Aunque sigue existiendo la posibilidad de que alguno de los *taggers* se haya equivocado, y si en todos los atributos las etiquetas son iguales, exceptuando los generalizados, entonces estamos construyendo una etiqueta válida pero incorrecta (recordemos que un *tagger* está entrenado para inferir el atributo número y el otro género). Como vemos el único tipo de error que podemos detectar es la invalidez de las etiquetas al combinarlas.

La única información con la que contamos para combinar las etiquetas en los métodos que implementamos es la salida de los *taggers*, por lo tanto si la salida de los *taggers* es errónea no podremos corregir los errores. Podríamos modificar los *taggers* para que la salida sea la etiqueta inferida y el contexto que utilizó para inferir esta etiqueta: si dudamos de la etiqueta inferida se podría reutilizar el contexto con algún método de clasificación para inferir

otro resultado, quizás correcto.

Bibliografía

[1] S. Abney. Part-of-speech tagging and partial parsing, 1996.

[2] S. P. Abney. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer, Dordrecht, 1991.

[3] Eugene Charniak. Statistical language learning, 1993.

[4] T. Kudo and Y. Matsumoto. Use of support vector learning for chunk identification, 2000.

[5] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, 1999.

[6] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[7] Miles Osborne. Shallow parsing as part-of-speech tagging. In Claire Cardie, Walter Daelemans, Claire Nédellec, and Erik Tjong Kim Sang, editors, *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, Lisbon, 2000*, pages 145–147. Association for Computational Linguistics, Somerset, New Jersey, 2000.

[8] M. Palomar, M. Civit, A. Díaz, L. Moreno, E. Bisbal, M. Aranzabe, A. Ageno, M. A. Martí, and B.Ñavarro. 3lb: Construcción de una base de datos de árboles sintáctico-semánticos para el catalán, euskera y español. In *XX Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN)*, pages 81–88, 2004.

[9] Lance Ramshaw and Mitch Marcus. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church,

editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[10] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996.

[11] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

[12] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000.

[13] C.J. van Rijsbergen. Information retrieval, 1975.