

Monografías de la Revista

Neolnstrumenta

Revista del Laboratorio de Ingeniería Documental

Año 2014

PROCESAMIENTO DEL LENGUAJE NATURAL

Fuensanta M^a Guerrero Carmona
Manuel Marcos Aldón

Contenido

PARTE I. AUTÓMATAS FINITOS, PROCESAMIENTO DE UNIDADES MORFOLÓGICO-LÉXICAS Y ETIQUETADO SINTÁCTICO	5
1. Introducción.....	6
2. Expresiones Regulares y Autómatas Finitos.....	7
2.1 Expresiones regulares.....	7
2.2 Autómatas de estados finitos.....	7
2.3 Reconocimiento de cadenas	9
2.4 Autómatas de estados finitos no deterministas (NFSAs)	12
3. Procesamiento de unidades morfológico-léxicas	14
3.1 Estudio de la morfología	14
3.2 Lexicón de estado finito	19
3.3 Transductores de estado finito.....	20
3.4 Análisis morfológico y reglas ortográficas con transductores léxicos	21
3.5 Transductores de léxico libre: lematizador de Porter	22
3.6 Tokenización	22
4. N-Gramas.....	23
4.1 N-Gramas simples	23
4.2 Corpus de entrenamiento y de test	24
4.3 Smoothing (suavizado).....	25
4.4 Interpolación y Backoff.....	26
4.5 Análisis morfológico-léxico aplicado a motores de búsqueda textuales.....	26
5. Etiquetado.....	27
5.1 Conjunto de etiquetas para el inglés.....	27
5.2 Etiquetado <i>Part-of-Speech</i> o POS-tagging.....	28
5.3 POS-tagging basado en reglas.....	28
5.4 POS-tagging estocástico.....	29
5.5 Etiquetado basado en transformación (Transformation-Based tagging)	29
6. Conclusiones	30
PARTE II. GRAMÁTICAS DE CONTEXTO LIBRE, PARSING, UNIFICACIÓN DE RASGOS Y SEMÁNTICA Y ANÁLISIS SEMÁNTICO.....	33
7. Introducción.....	34
8. Gramáticas de contexto libre para el análisis del lenguaje natural.....	35
8.2 Reglas gramaticales.....	36

8.3	Treebanks	38
8.4	Equivalencia gramatical y forma normal	38
8.5	Estado finito y gramáticas de contexto libre	39
8.6	Gramáticas de dependencia.....	40
9.	 Análisis sintáctico	40
9.1	El análisis como búsqueda	40
9.2	Ambigüedad	42
9.3	Buscar en la superficie de la ambigüedad	42
9.4	Métodos de análisis de programación dinámica.....	43
9.5	Análisis parcial.....	45
10.	 Unificación de rasgos	46
10.1	Estructuras de rasgos	46
10.2	Unificación de estructuras de rasgos	48
10.3	Estructuras de rasgos en gramática	48
10.4	Implementación de la unificación	50
10.5	Análisis con restricciones de unificación	52
11.	 Semántica y análisis semántico	52
11.1	Representación del significado.....	52
11.2	Semántica computacional.....	59
11.3	Semántica léxica.....	63
11.4	Semántica léxica computacional	66
12.	 Conclusiones	66
	Glosario de términos	69
	PARTE III. DISCURSO, EXTRACCIÓN DE INFORMACIÓN Y RESÚMENES.....	72
13.	 Introducción.....	73
14.	 Discurso.....	74
14.1	Segmentación del discurso	74
14.2	Referencias	74
15.	 Extracción de información	75
15.1	Reconocimiento de entidades nombradas	75
15.2	Detección y clasificación de relaciones entre entidades.....	76
15.3	Procesamiento temporal y de eventos	77
16.	 Question Answering (Búsqueda de Respuestas).....	78
17.	 Sistemas de extracción de resúmenes	79

17.1	Componentes de un algoritmo de resumen	80
17.2	Evaluación de los sistemas de resúmenes	82
18.	Conclusiones	83
	Bibliografía	85

**PARTE I. AUTÓMATAS FINITOS,
PROCESAMIENTO DE UNIDADES
MORFOLÓGICO-LÉXICAS Y ETIQUETADO
SINTÁCTICO**

Resumen. El incremento de los recursos disponibles a través de la Web y la disponibilidad de acceso móvil inalámbrico han colocado a las aplicaciones del procesamiento de la voz y del lenguaje en el centro de atención de las investigaciones tecnológicas. El procesamiento del lenguaje natural (PLN) es una disciplina que diseña, implementa e investiga sobre distintos mecanismos computacionales para establecer una comunicación efectiva entre los ordenadores y las personas. Lo que distingue el procesamiento del lenguaje de otros sistemas de procesamiento de datos es que hace uso del *conocimiento del lenguaje* en todas sus facetas: fonética, fonológica, morfológica, sintáctica, semántica, pragmática y discursiva. Este trabajo se centra en el estudio de modelos y algoritmos que emplea el PLN, como las gramáticas regulares, los autómatas finitos y los transductores. Cada uno de estos modelos se puede ampliar con sistemas probabilísticos, cuya ventaja principal es su capacidad para resolver problemas de ambigüedad. Se analizan distintos procesos de tratamiento de unidades morfológico-léxicas que sirve de apoyo al etiquetado gramatical.

1. Introducción

El incremento de los recursos informáticos disponibles, con ordenadores cada vez más potentes, el aumento de la Web como fuente masiva de información y la creciente disponibilidad de acceso móvil inalámbrico hacen que las aplicaciones de procesamiento de la voz y del lenguaje sea el centro de atención de la tecnología.

Diversos modelos e investigaciones se han ido produciendo a lo largo de los años desde que, al terminar la II Guerra Mundial, se crearon los primeros paradigmas: los autómatas con modelos probabilísticos, y la teoría de la información. Los modelos simbólico y estocástico dieron paso a otros basados en la lógica, en la compresión del lenguaje natural, o en el modelo del discurso. Actualmente, gracias a los experimentos y estudios llevados a cabo y a la disponibilidad de sistemas informáticos de alto rendimiento, se ha producido un gran avance en la utilización de modelos probabilísticos y estadísticos en el área de la comprensión del procesamiento del lenguaje humano. A continuación, en el punto 2 se va a profundizar en el análisis de las expresiones regulares y los autómatas finitos, tanto en su versión determinista como no determinista, así como en el estudio de la relación que existe entre ellos. En el punto 3 se estudian diversas técnicas de procesamiento de las unidades morfológico-léxicas, tanto para el idioma inglés como para el castellano. Se analiza también el uso de transductores de estado finito para lematizar palabras. En el apartado 4 se detallan varios métodos para identificar *N*-gramas en un texto. Estas técnicas utilizan corpus de entrenamiento y de pruebas para calcular probabilidades que permiten predecir el siguiente elemento en una secuencia de palabras. El procesamiento del lenguaje con cualquiera de los modelos mencionados (autómatas finitos, transductores y *N*-gramas) implica una búsqueda a través de un espacio de estados que representan hipótesis sobre una entrada. Para las tareas no probabilísticas se utilizan algoritmos de grafos tales

como *búsqueda primero en profundidad* (depth-first). Para las tareas probabilísticas se utilizan variantes heurísticas, tales como *búsqueda primero el mejor* (best-first) y A*, empleando algoritmos de programación dinámica para que sean tratables computacionalmente. La sección 5 se centra en el etiquetado gramatical. Se estudian varios sistemas para etiquetar las palabras, utilizados tanto en algoritmos de desambiguación como en la detección de nombres, fechas u otras entidades. Finalmente, en el apartado 6 de este trabajo se presentan las conclusiones del estudio realizado.

2. Expresiones Regulares y Autómatas Finitos

2.1 Expresiones regulares

Una expresión regular es la notación estándar para la caracterización de una secuencia de caracteres (letras, números, espacios, tabuladores y signos de puntuación). Esta sucesión de símbolos forma un patrón. Se utilizan en la búsqueda de cadenas, ya que ofrecen una manera declarativa de expresarlas. Para realizar dichas búsquedas se requiere:

- un patrón: indica qué es lo que se busca
- un corpus: señala dónde se busca

El patrón se construye con diversos operadores. Los básicos, ordenados de mayor a menor precedencia, son:

- Paréntesis ()
- Cuantificadores * + ? { }
- Secuencias y anclas ^ \$
- Disyunción |

Las expresiones regulares son sensibles a las mayúsculas y minúsculas, y se utilizan los corchetes para indicar un rango de caracteres. Con el conjunto de todos los operadores básicos y avanzados se puede representar cualquier cadena de caracteres del lenguaje, describiendo de forma simple y precisa lenguajes regulares.

2.2 Autómatas de estados finitos

Una máquina de estados es un modelo de comportamiento de un sistema, que tiene una serie de entradas y salidas, y dónde las salidas dependen no sólo de las señales de entrada actuales, sino también de las anteriores. Este tipo de máquinas se denomina

máquina de estados finitos si el conjunto de estados de la máquina tiene un límite y, por lo tanto, se puede modelar en un ordenador.

Un autómata de estados finitos (o simplemente autómata finito) es similar a una máquina de estados finitos, excepto que el autómata tiene estados de aceptación y de rechazo en lugar de una salida.

De hecho, se puede definir a los autómatas finitos como el modelo matemático de los sistemas que presentan las siguientes características:

- En cada momento el sistema se encuentra en un estado, y el conjunto total de estados diferentes en los que se puede encontrar el sistema es finito.
- Pueden responder a un número finito de sucesos distintos.
- El estado en el que se encuentra el sistema es consecuencia de toda la información relacionada con los sucesos previos.
- La respuesta a un suceso sólo se define según dicho suceso y el estado actual en el que se encuentra el sistema.

Una expresión regular es una forma de especificar un autómata finito. Cualquier expresión regular se puede implementar como un autómata finito. Análogamente, cualquier autómata finito se puede describir con una expresión regular. Tanto las expresiones regulares como los autómatas finitos se pueden utilizar para determinar lenguajes regulares. Por otra parte, las gramáticas regulares son aquellas que generan lenguajes regulares. En consecuencia, se establece una estrecha relación entre las expresiones regulares y los autómatas finitos, que se demuestra por el hecho de que para cualquier expresión regular existe un autómata finito que reconoce el lenguaje que describe, y cada autómata finito reconoce un lenguaje que se puede determinar con una expresión regular.

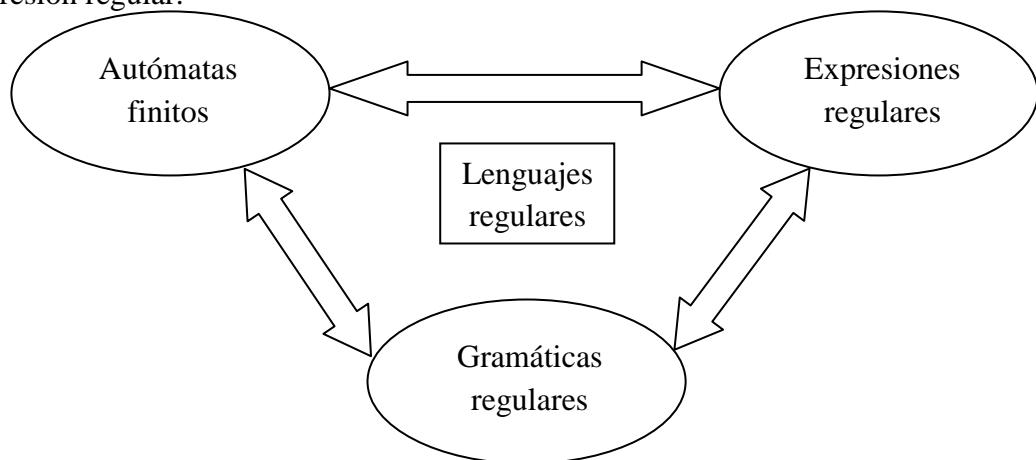


Figura 1. Relación entre autómatas finitos y expresiones regulares

2.3 Reconocimiento de cadenas

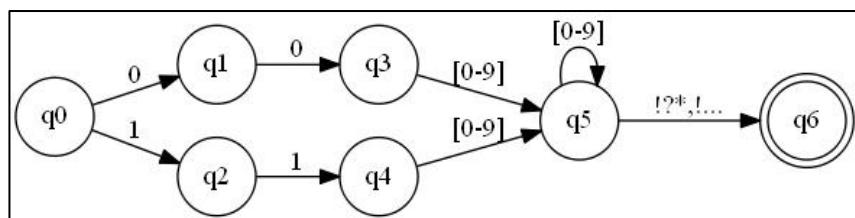
Los autómatas finitos, como se acaba de indicar, se utilizan para reconocer lenguajes regulares, que corresponde a los lenguajes formales más simples. Un lenguaje formal es un conjunto de cadenas compuestas por símbolos que pertenecen a un conjunto finito, llamado alfabeto. No son lo mismo que los lenguajes naturales, que son los utilizados por las personas, pero se puede utilizar un lenguaje formal para modelar parte de un lenguaje natural.

Se detallan a continuación algunos ejemplos de expresiones regulares, mostrando también su correspondiente autómata finito:

- Supongamos una cadena que comienza con dos ceros consecutivos ó dos unos consecutivos, seguidos por uno ó más números cualesquiera y termina con un símbolo no alfanumérico.

Expresión regular: $(00|11)d^+!W$

Autómata finito equivalente:



Cadenas aceptadas:

00112233434546645!

114377332423.

Cadenas rechazadas:

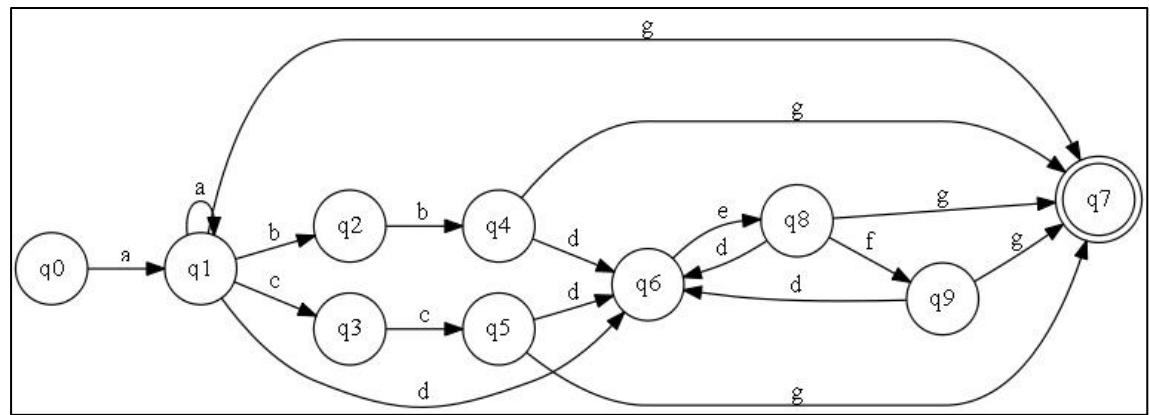
0112342323! (comienza por un solo cero, aunque después vayan dos “1”)

006454b. (contiene una letra)

- Cadena que comienza con una “a” ó más, seguida por la cadena “bb” ó “cc” opcional, y después por la cadena “def” o “de” opcional, y termina con una “g”.

Expresión regular: $a+(bb|cc)?(def?)^*g$

Autómata finito equivalente:



Cadenas aceptadas:

aaabbdeg

adefg

Cadenas rechazadas:

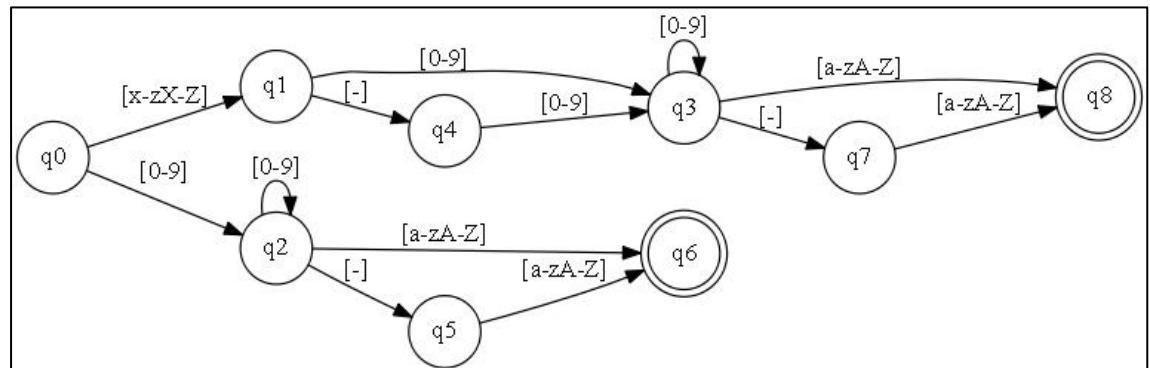
bbdefg (no comienza con “a”)

aabbdefhg (contiene una “h”)

- Número de DNI o NIE:

Expresión regular: $(\text{d}+[-]?\text{[a-zA-Z]})([\text{x-zA-Z}][-]?\text{d}+[-]?\text{[a-zA-Z]})$

Autómata finito:



Cadenas aceptadas:

30508571z

X-2345678-V

Cadenas rechazadas:

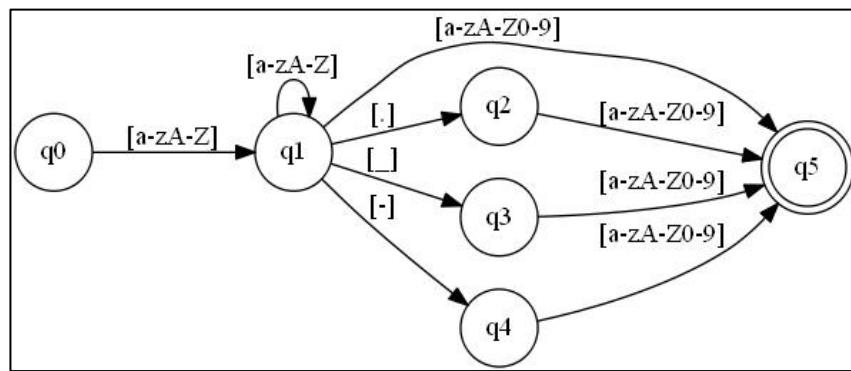
a2322356 (comienza con una letra que no está en el intervalo [x-z])

B-3546335! (contiene el símbolo !)

- Nombre de usuario. Debe comenzar con una o más letras, y puede contener números. También puede tener guiones bajos, puntos o guiones, pero no al comienzo o final, ni puede tener más de uno de estos caracteres juntos.

Expresión regular: / [a-zA-Z]+(\.|_|-)?([a-zA-Z0-9])+ /

Autómata finito:



Cadenas aceptadas:

tanti.guerrero24

pp_sanchez

Cadenas rechazadas:

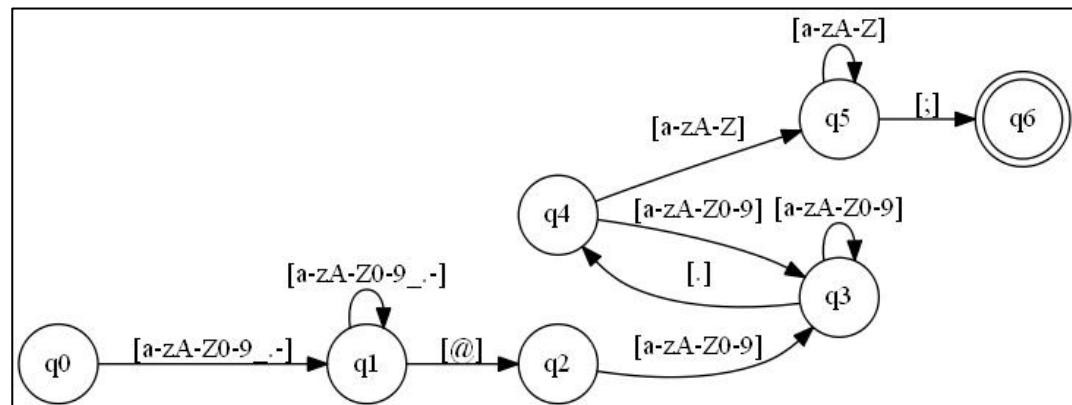
fuen.-guerrero12 (tiene un punto y un guión seguidos)

2fuen.guerrero (comienza con un número)

- Dirección de correo electrónico:

Expresión regular: [a-zA-Z0-9_.-]+@[a-zA-Z0-9_.-]+\.[a-zA-Z]+[;]

Autómata finito:



Cadenas aceptadas:

fuen.guerrero@gmail.com;

anamigc_34@yahoo.es;

Cadenas rechazadas:

fuen.guerrero*@gmail.com; (tiene un asterisco)

fuen.guerrero@gmail..com; (tiene 2 puntos seguidos después de la @)

Para comprobar la corrección y el funcionamiento en la aceptación de cadenas de las expresiones regulares anteriores se ha utilizado *regexpal* [9]. Para realizar los gráficos correspondientes a los autómatas finitos se ha utilizado el programa Graphviz [2].

2.4 Autómatas de estados finitos no deterministas (NFSAs)

Son autómatas finitos con puntos de decisión, en los que de antemano no se sabe cuál es el estado siguiente, a diferencia de los deterministas en los que, durante el proceso de reconocimiento, su comportamiento está totalmente definido por el estado en que se encuentra y el símbolo que se está considerando. Otro tipo de NFSA es aquel que tiene arcos sin símbolo (llamados ϵ -transitions o transiciones vacías), en los que se debe decidir entre seguir la ϵ -transition o el arco siguiente.

Para decidir qué arco elegir en los NFSAs se puede optar por una de estas tres soluciones:

- **Backup:** consiste en poner un marcador para almacenar la posición de la entrada y el estado del autómata en ese momento, para así poder volver hacia atrás e intentar un nuevo camino en caso de haber hecho una elección incorrecta.
- **Look-ahead:** se basa en observar la cadena y ver cuál es el siguiente elemento de la entrada para tomar una decisión sobre qué camino seguir.
- **Parallelism:** al llegar a un punto de elección, se consideran los caminos alternativos paralelamente.

El enfoque de Backup sugiere que podemos elegir de forma irreflexiva caminos que pueden llevarnos a callejones sin salida, puesto que podemos volver a alternativas no probadas. Hay dos claves en este enfoque: necesitamos recordar todas las alternativas para cada punto de elección, y necesitamos almacenar información suficiente sobre cada alternativa para poder regresar cuando sea necesario. El algoritmo utilizado con los NFSAs sólo rechaza la cadena cuando todas las posibles opciones han sido examinadas y no se ha llegado a un estado de aceptación.

2.4.1 Reconocimiento como búsqueda

Si después de recorrer los posibles caminos se llega a un estado de aceptación, la cadena es aceptada, si no, se rechaza. Esto es posible gracias al mecanismo de la agenda, que en cada iteración selecciona un camino a explorar y guarda un registro de los caminos no explorados. Este tipo de algoritmos, que sistemáticamente buscan soluciones, se conocen como algoritmos de **búsqueda en el espacio de estados** (state-space search). En ellos la definición del problema crea un espacio de soluciones posibles, explora este espacio, y devuelve una respuesta cuando encuentra una solución, o rechaza la entrada cuando el espacio ha sido totalmente explorado.

Su efectividad depende del orden en el que se consideran los estados del espacio. Se puede crear un orden tal que el siguiente camino a considerar sea el creado más recientemente (pila). Esta es una estrategia de búsqueda **primero en profundidad** (depth-first search) o **último en entrar primero en salir** (LIFO).

Otra forma de ordenar los estados del espacio de búsqueda es considerar el orden en que fueron creados, colocando los nuevos estados al final de la agenda, y siendo el siguiente el estado que está el primero (el que lleva más tiempo en la cola). Esta estrategia se llama búsqueda **primero en anchura** (breadth-first search) o **primero en entrar primero en salir** (FIFO).

Para problemas pequeños estas dos estrategias son adecuadas, pero se prefiere *depth-first search* porque hace un uso más eficiente de la memoria. Para problemas más complicados se utilizan otras técnicas tales como **programación dinámica** o **A***.

2.4.2 Relación entre DFSA y NFSAs

Se podría pensar que los NFSAs son más potentes que los DFSA por el hecho de tener algunas características adicionales, pero no es así, sino que para cada NFSAs existe un DFSA equivalente. Hay un algoritmo para convertir un NFSAs en un DFSA, aunque el número de estados en el autómata no determinista puede ser mucho mayor debido a las transiciones vacías.

No existe gran diferencia entre estos dos tipos de autómatas. En los deterministas, cada transición está totalmente especificada, mientras que los no deterministas pueden tener estados en los que se debe decidir qué camino tomar. Por lo tanto, cuando se utilizan para el reconocimiento de cadenas, los NFSAs al llegar a un punto en el que deben seleccionar un camino, que puede ser erróneo y requiere retroceder para seguir otro, llevan consigo la necesidad de realizar una búsqueda en el espacio de estados.

Además, como se demuestra en [7], todas las expresiones regulares se pueden convertir en un autómata finito no determinista equivalente, y viceversa. No obstante, la conversión de un autómata finito no determinista en su expresión regular equivalente a veces conlleva un incremento en la complejidad de su descripción.

2.4.3 Lenguajes regulares y FSAs

Los lenguajes que se pueden definir utilizando expresiones regulares son del mismo tipo que los que se pueden caracterizar por los autómatas finitos (deterministas o no deterministas). Todos los operadores de expresiones regulares (excepto la memoria) se pueden implementar con estas tres operaciones: concatenación, unión o disyunción (\cup), y el cierre de Kleene. Además, los lenguajes regulares tienen estas operaciones: intersección, diferencia, complementación e inversión.

3. Procesamiento de unidades morfológico-léxicas

3.1 Estudio de la morfología

La morfología es el estudio de la forma en la que se construyen las palabras a partir de las unidades más pequeñas con significado, los morfemas. Hay dos tipos de morfemas: raíz (stem) y afijo (affix). Se puede decir que la raíz es el morfema principal de la palabra, la que le da su significado fundamental, y los afijos son significados adicionales.

Los afijos se dividen en prefijos (se colocan delante de la raíz), sufijos (van detrás de la raíz), infijos (en medio, por ejemplo en tagalog) y circunfijos (delante y detrás, como en alemán, para formar un tiempo verbal).

Hay muchas formas de combinar morfemas para crear palabras. Las más comunes son:

- **Inflexión:** combinación de la raíz de una palabra con un morfema gramatical que produce una palabra de la misma clase gramatical que la original.
- **Derivación:** unión de la raíz de una palabra con un morfema gramatical, dando como resultado una palabra de diferente clase gramatical.
- **Composición:** combinación de varias raíces juntas en una misma palabra.
- **Uso de clíticos:** concatenación de la raíz de una palabra con un clítico, que es un morfema que actúa sintácticamente como una palabra, pero en forma reducida.

3.1.1 Morfología del idioma inglés

En inglés los morfemas pueden ser libres o ligados. Los libres tienen significado independiente, mientras que los ligados necesitan estar unidos a otros morfemas para completar su significado. Una clasificación podría ser esta:

- Libres:
 - con contenido: boy, run, green, paper, quick, large, now, etc.
 - de función: of, and, the, to, a, but, some, that, there, etc.
- Ligados (son los afijos):
 - flexivos: son sufijos, y no cambian la clase gramatical de la palabra original
 - derivados: pueden ser prefijos o sufijos, y suelen cambiar la clase gramatical de la palabra original

En inglés, una palabra puede tener más de un afijo, pero normalmente no más de 4 ó 5.

Para los ejemplos que aparecen en las tablas se han utilizado *snowball* [13] y *FreeLing* [12].

Prefijos	Sufijos	Infijos	Circunfijos	Enclíticos
disagree	approachable	cupsful	unbelievably	we've
impossible	golden	narcotic	unconsciousness	he'll
incorrect	designer	coniferous	unlikely	I'm
mislead	helpful	astrology	rewrites	they're
preview	reversible		enlighten	my's
rename	action		embolden	uncle's
uncertain	careless			
international	goodness			

- Morfología flexiva: sólo los nombres, verbos, y algunos adjetivos se pueden declinar. Los nombres tienen dos tipos de inflexión: un afijo (sufijo) para el plural y otro para el posesivo. La inflexión verbal es algo más complicada. Los verbos pueden ser principales, modales (can, will, should) y primarios (be, have, do). Dentro de los principales los hay regulares e irregulares. Los regulares tienen cuatro formas morfológicas (raíz, -s, -ing participio o gerundio, y -ed o pasado). Los irregulares tienen cinco formas morfológicas (raíz, -s, -ing participio o gerundio, pasado y participio). Los adjetivos se declinan en el comparativo y superlativo.

Nombre s	Verbos				Adjetivos	
	-s	ing	pasado	participi o	comparativ o	superlativ o
girls	writes	writing	wrote	written	cheaper	cheapest
books	brings	bringing	brought	brought	crazier	craziest
parties	concern s	concernin g	concerne d	concerne d	longer	longest
buses	damages	damaging	damaged	damaged	bigger	biggest
potatoes	tries	trying	tried	tried	busier	busiest
babies						
father's						
families'						

- Morfología derivativa: es más compleja que la inflexión. El tipo de derivación más común es la formación de nuevos nombres, a menudo a partir de verbos o adjetivos. Este proceso se llama nominalización. También se pueden衍生 adjetivos a partir de nombres y verbos. Como se ha indicado anteriormente, con el uso de prefijos normalmente no altera la clase gramatical de la palabra raíz, y con los sufijos suele cambiar tanto el significado como la clase gramatical de la palabra.

Sufijos			
adj. → nombre	sadness	darkness	kindness
adj. → verbo	materialise	vocalize	privatize
adj. → adj.	selfish	greenish	
adj. → adverb.	stately	monthly	
nombre → adj.	functional	professional	universal
nombre → verbo	clarify	specify	
verbo → adj.	readable	helpful	usable
verbo → nombre	relevance	worker	governance

Prefijos		
decode	impossible	prefix
disagree	mislead	react
illegal	nonsense	undo

- Concordancia: el sujeto y el verbo deben concordar en persona y número; y el nombre y el pronombre en número, persona y género.

Concordancias	
sujeto-verbo	The boy reads mistery stories.
sujeto-verbo	Hight levels of mercury occur in some fish.
sujeto-verbo	Water in the fuel lines causes an engine to stall.
nombre-antecedente	Every man over 44 years of age should know the level of his prostate-specific antigen.
nombre-antecedente	Genes are the microscopic parts of a living organism that determine its structure and functions.
nombre-antecedente	Teachers can create podcasts for each child , helping him or her learn how to work in groups as well as plan his or her own project.

3.1.2 Morfología del idioma castellano

En castellano, un morfema es la unidad mínima analizable que posee significado gramatical. Se dividen en:

- Independientes o clíticos:
 - libres: flor, luz, mar, camión, reloj, etc.
 - determinantes: el, la, una, estos, le, mi, vuestras, los, se, tres, algún, qué, etc.
 - preposiciones: a, por, durante, con, entre, etc.
 - conjunciones: antes de que, y, ni, por lo tanto, que, pero, sin embargo, etc.
- Dependientes o ligados (son los afijos):
 - flexivos: indican relaciones gramaticales entre los participantes de una acción verbal o expresión nominal
 - derivativos: añaden matices al significado

Para los ejemplos que aparecen en las tablas se ha utilizado el *lematizador* de la Universidad de Las Palmas [16] y *Snowball* [13] y *FreeLing* [12].

Prefijos	Sufijos	Infijos	Circunfijos	Enclíticos
hiperactivo	neurálgia	panadero	anaranjar	ponerlo
anticuerpo	aerodromo	tubular	descascarillar	sigueme
democracia	claustrofobia	soluble	ensuciar	peinarse
interceder	geología	picotazo	aburguesado	dile
infarrojo	pastor	matorral	aparcar	dejarnos
superdotado	filosofía	espionaje		guárdate
omnipresente	tolerancia	ventolera		sáltalas
amorfo	clorofila	laborioso		tragatelo

- Morfología flexiva: puede ser verbal (conjugación) o nominal (declinación). Esta última se aplica a sustantivos, pronombres y adjetivos. No implica cambios semánticos, sólo gramaticales. Los verbos tienen afijos que indican tiempo, modo, aspecto, voz, número y persona. En el caso de los nombres, existen afijos para señalar el género, número y, a veces, caso.

Nominal			Verbal				
sustantivos		pronom.	adjetivos	tiempo-modo-aspecto	número-persona	voz	
M	niño	este	atento			activa	pasiva
F	mesa	ella	atrevida	saludaran	saltas	leyó	fue leído
N	importante	esto	brillante	salíamos	subimos	prepara	es preparada
P	árboles	suyos	educados	esperando	estudiais	otorgará	será otorgado
S	botella	vuestro	poderoso	prohibido	aparecen		

- Morfología derivativa: se construyen nuevas palabras a partir de otras. Se puede hacer añadiendo prefijos o sufijos. Los prefijos no cambian la categoría gramatical de una palabra, mientras que los sufijos pueden cambiarla. Otra forma de construir palabras nuevas es la composición, en la que dos o más palabras forman conjuntamente otra distinta: sacapuntas, pelirrojo, agridulce, drogadicto, etc.

Sufijos			
adj. → sustantivo	sumisión	regularidad	altura
verbo → sustant.	comida	instrumentalización	estacionamiento
sustant. → sustant.	periodismo	casaamiento	maquinaria
adj. → verbo	institucionalizar	blanquear	dormitar
sustant. → verbo	fotografiar	mapear	
sustant. → adj.	climático	abrigado	universal
verbo → adj.	bailable	vencido	lucido

Prefijos		
deslucir	insumiso	preacuerdo
irregular	supermáquina	microclima
restablecer	semiautomático	extraordinario

- Concordancia: existe concordancia de género, número, persona, tiempo y modo.

Concordancias	
nominal (género y número)	El niño rubio levantó la mano. Les di tu teléfono a mis padres .
verbal (número, persona, tiempo y modo)	Los gatos comían pescado. El joven baila muy bien. Juan quiere que Luis venga . Juan quiso que Luis viniera .

3.2 Lexicón de estado finito

Un lexicón es un repositorio de palabras. El más simple consiste en una lista explícita de cada palabra del lenguaje (incluyendo abreviaturas y nombres propios).

Ya que resulta imposible listar todas las palabras del lenguaje, los lexicones computacionales normalmente son estructurados con una lista de cada raíz y afijo del lenguaje, junto con una representación de los morphotactics que nos dicen cómo pueden encajar. Hay muchas formas de modelar morphotactics; una de las más comunes es el autómata de estados finitos, con un estado inicial q_0 , transiciones a otros estados intermedios y un estado final.

La morfología derivacional es mucho más compleja que la flexiva en inglés, por lo que el autómata para modelar la derivación tiende a ser bastante complejo en inglés.

3.3 Transductores de estado finito

Un transductor de estado finito (FST), es un tipo de autómata finito que enlaza dos conjuntos de símbolos. Se puede decir que un FST es como una máquina que lee una cadena y genera otra.

Para el análisis morfológico, y para muchas otras aplicaciones de PLN, el FST se asimila a un traductor que coge como entrada una cadena de letras y produce como salida una cadena de morfemas.

Los transductores secuenciales son un subtipo de transductores que son deterministas en sus entradas y, por lo tanto, pueden convertirse en su homólogo determinista. Los transductores secuenciales pueden tener símbolos ϵ en la cadena de salida, pero no en la de entrada. Por tanto no son necesariamente secuenciales en su salida. Puesto que el inverso de un transductor secuencial puede no ser secuencial, se debe especificar la dirección de transducción. Un transductor subsecuencial es aquel donde los símbolos de salida se generan sólo cuando se han visto suficientes símbolos en la entrada para garantizar una salida correcta. Se puede decir que es un transductor secuencial ampliado para permitir una cadena de salida adicional. Lo que hace importantes a los transductores secuenciales y sub-secuenciales es su eficacia; ya que son deterministas en su entrada, pueden ser procesados proporcionalmente al número de símbolos de la entrada en lugar de hacerlo según el número de estados, que es mucho más grande. Otra ventaja de los transductores sub-secuenciales es que hay algoritmos eficientes para hacerlos deterministas, y también para minimizarlos. Pero ninguno puede encargarse de la ambigüedad, propiedad fundamental del lenguaje natural. Para tratar algunas ambigüedades existe una generalización de los transductores subsecuenciales, el transductor p -subsecuencial, que permite que p cadenas de salida sean asociadas con cada estado final. Con este tipo de transductores se puede limitar la ambigüedad en diccionarios, compilación de reglas morfológicas y fonológicas y restricciones sintácticas [10].

3.4 Análisis morfológico y reglas ortográficas con transductores léxicos

Para el análisis morfológico se representa una palabra como una correspondencia entre el nivel léxico (concatenación de morfemas) y el nivel superficial (concatenación de letras según la ortografía). Pero a veces se deben emplear reglas ortográficas para formar el plural o un tiempo verbal en palabras irregulares, para lo que se utiliza un nivel intermedio para especificar la regla apropiada.

Un transductor léxico relaciona el nivel léxico (raíces de las palabras y características morfológicas) con el nivel intermedio, que representa una concatenación de morfemas. Después varios transductores, cada uno representando una regla ortográfica, se ejecutan en paralelo para enlazar el nivel intermedio con el superficial. Esto se realiza en forma de cascada o secuencia.

Esta ejecución en cascada se puede llevar a cabo de forma más eficiente utilizando la composición e intersección de los transductores. La composición se define así:

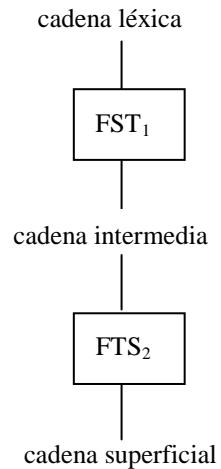
Dados dos transductores, F y G , existe un transductor $F \circ G$ tal que, siendo x una entrada de F cuya salida es y , suministramos como entrada a G esta y , nos da como salida z . Es decir, dadas dos relaciones F y G , $(x,z) \in F \circ G$ cuando existe alguna y tal que $(x,y) \in F$ y $(y,z) \in G$.

La intersección se define de esta forma:

Dados dos transductores F y G , la intersección define una relación $R(F \wedge G)$ tal que $R(x,y)$ si y sólo si $F(x,y)$ y $G(x,y)$.

En Mohri [11] se señala que la composición de transductores consiste en combinar varios niveles de representación, y que se puede implementar en sistemas de reconocimiento de voz.

Como se ha indicado anteriormente, se utilizan estas propiedades de los transductores para tratar el problema de la ambigüedad de una forma eficiente, ya que aportan información adicional al transductor. Estas propiedades permiten al transductor buscar entre las posibles salidas para seleccionar la más apropiada según la entrada y el contexto. Para que sean eficaces, se construyen transductores minimizados y deterministas.



3.5 Transductores de léxico libre: lematizador de Porter

A veces, los sistemas de recuperación de información ejecutan un lematizador sobre la consulta y las palabras que hay en el documento, eliminando los sufijos. Uno de los algoritmos para lematizar más utilizados, simple y eficiente, es el de Porter, basado en una serie de reglas muy simples de reescritura en cascada. Como este tipo de reglas pueden ser fácilmente implementadas como un transductor de estado finito, el algoritmo de Porter se puede asimilar a un lematizador para un transductor de léxico libre. Pero este tipo de lematizadores pueden producir errores de comisión (organization-organ) o de omisión (European-Europe).

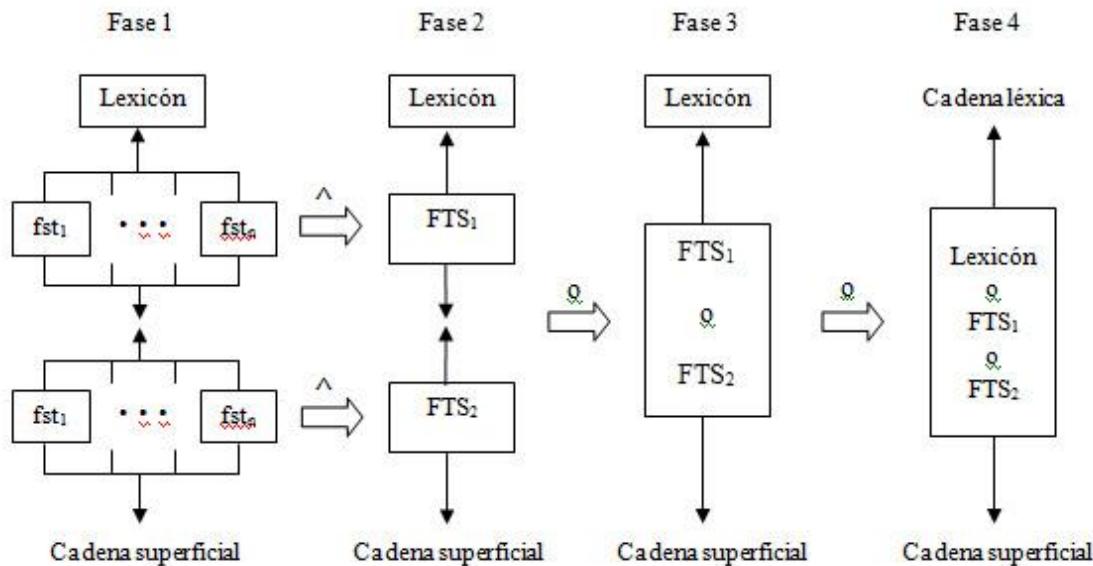


Figura 2. Implementación del algoritmo de Porter con transductores

En la Figura 2 [8] se muestra la manera en que se utilizan las propiedades de composición (símbolo \circ) e intersección (símbolo \wedge) de los transductores para construir un lematizador utilizando el algoritmo de Porter. En cada transductor se puede implementar una de las reglas del algoritmo. La página de *snowball* [13] contiene lematizadores para 12 idiomas diferentes, incluido el castellano.

3.6 Tokenización

Igual que las palabras se dividen en morfemas (lematización), el texto se divide en frases y palabras (tokenización). Para realizar esta división, tanto en inglés como en castellano, se utilizan los espacios en blanco y los signos de puntuación. Pero en las abreviaturas, las cifras numéricas, los enclíticos o las direcciones Web no basta con estos signos para llevar a cabo un tokenización correcta.

Otro problema lo presentan las expresiones compuestas de varias palabras, como *New York* o *Ministerio de Justicia*. La *detección de entidades nombradas* es el proceso que se encarga de detectar fechas, nombres propios y nombres de organizaciones.

En la tokenización de frases el problema lo presenta el punto, ya que este signo ortográfico puede indicar el final de una frase o el final de una abreviatura, o incluso formar parte de una URL. Para solucionar este inconveniente se suelen utilizar diccionarios de abreviaturas, además de técnicas de aprendizaje automático.

4. N-Gramas

Una de las herramientas más importantes para el procesamiento de la voz y del lenguaje es el modelo de *N*-gramas, que asigna una probabilidad condicional a las palabras que posiblemente sean las siguientes en una secuencia. Los modelos de *N*-gramas también son fundamentales en la traducción automática, ya que pueden servir para indicar la traducción más probable de un grupo de palabras o *gramas*.

En procesamiento del lenguaje natural se utilizan los *N*-gramas en tareas como etiquetado gramatical, generación de lenguaje natural, similitud de palabras, identificación de autores y extracción de opiniones.

El cálculo de probabilidades se basa en contar cosas, en nuestro caso, palabras. Si conocemos la frecuencia de que *N* palabras aparezcan juntas, podremos calcular la probabilidad de que, si aparecen las *N*-1 primeras, la siguiente sea la palabra colocada en el lugar *N*.

4.1 N-Gramas simples

Se deben utilizar formas inteligentes de calcular la probabilidad de una palabra *w* dada una historia *h*, o la probabilidad de una secuencia completa de palabras *W*. Una de ellas es aplicar la *regla de la cadena de probabilidad*, que muestra el enlace entre el cálculo de la probabilidad conjunta de una secuencia y el cálculo de la probabilidad condicional de una palabra dadas las anteriores. Pero el lenguaje es creativo, y aparecen frases nuevas que no se habían utilizado anteriormente. En consecuencia, las probabilidades no se pueden calcular exactamente, por lo que se utiliza la aproximación proporcionada por los modelos de *N*-gramas. En estos modelos, se miran las *N*-1 palabras previas para calcular la probabilidad de una palabra.

Para calcular esta probabilidad, se utiliza la *estimación de probabilidad máxima*, normalizando los recuentos de un corpus, que entonces oscilarán entre 0 y 1:

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1})}{C(w_{n-N+1}^{n-1})}$$

4.2 Corpus de entrenamiento y de test

Para calcular las probabilidades de los modelos de N -gramas se divide el corpus en un grupo de entrenamiento (training set o corpus de entrenamiento) y otro de test (test set o corpus de test). Se calculan sobre el *training set* y su resultado se prueba calculando nuevas probabilidades en el *test set*. De esta forma se pueden comparar distintos modelos para saber cual obtiene mejores resultados.

La perplejidad mide la eficacia de un modelo en relación con su exactitud en el *test set*, es decir, el mejor modelo es el que consigue un mayor ajuste con los datos de prueba o el que realiza una mejor predicción de los detalles de dichos datos de prueba. En consecuencia, el mejor modelo asignará una probabilidad más alta a los datos del test y un valor de perplejidad más bajo, ya que es inversa a la probabilidad (se mostrará menos "sorprendido" si la probabilidad es alta). A veces se necesitan varios *test sets* para que las medidas sean más objetivas, por lo que se utiliza un *test set de desarrollo*.

Las frases del *training set* no pueden estar en el de *test set*, ya que si no se darían imprecisiones en el cálculo de la perplejidad. Además, el tamaño del *training set* debe ser tan grande como sea posible para tener muchos datos, pero no puede contener todos. Por otro lado, el *test set* tiene que ser lo más pequeño posible, pero teniendo en cuenta que debe proporcionar suficiente potencia estadística para medir la diferencia entre dos modelos. El corpus de datos se suele dividir de la siguiente manera: 80% para el *training set*, 10% para el *test set de desarrollo* y 10% para el *test set*.

Un *training set* lo suficientemente grande y un *test set* adecuadamente pequeño garantizan una correcta medida de la perplejidad. Un *training set* pequeño y un *test set* grande provocan peores resultados, ya que hay menos ejemplos disponibles para el entrenamiento y además la medida de la perplejidad es errónea, porque casi siempre será muy alta.

Otro aspecto a tener en cuenta es que el *training set* y el *test set* deben pertenecer al mismo género. No se deben coger textos mezclados procedentes de periódicos, ciencia ficción, conversaciones telefónicas y páginas Web, por ejemplo. Pero si la investigación que se va a llevar a cabo no tiene un dominio específico, se han de seleccionar corpus de entrenamiento balanceados, que incluyen textos de múltiples géneros.

A veces el estudio a realizar incluye un *vocabulario cerrado*, en el que conocemos todas las palabras que van a aparecer en los documentos y en el *test set*. En la práctica, esto es una simplificación, ya que pueden surgir palabras nuevas que no estaban en el vocabulario inicial. Por otra parte, un sistema con *vocabulario abierto* es aquel en el que se modelan las posibles palabras desconocidas en el *test set*, añadiendo una etiqueta <UNK> a cada una de ellas en el *training set*.

4.3 Smoothing (suavizado)

El proceso para calcular la estimación de probabilidad máxima de los parámetros de un modelo de N -grama presenta el problema de la escasez de los datos, ya que se realiza sobre un conjunto de datos de entrenamiento. En consecuencia, a veces se asigna una probabilidad de 0 a los N -gramas que no se encuentran en el *training set*, aunque en realidad no sea esta su probabilidad. Como resultado, también la medida de la perplejidad se ve afectada. Para resolver estos problemas se utilizan técnicas de *smoothing* o *suavizado*, que asignan una probabilidad distinta de cero a cualquier N -grama, incluso aunque no aparezca entre los datos de entrenamiento.

Existen varios algoritmos para realizar el *smoothing*:

- De Laplace: normaliza las probabilidades asumiendo que todos los N -gramas han aparecido al menos una vez en el *training set*, sumando 1 a los recuentos de todos los N -gramas. El problema de este algoritmo es que atribuye demasiada probabilidad a los N -gramas que no están en el corpus de entrenamiento. Por este motivo no se suele utilizar.
- Descuento de Good-Turing: utiliza los N -gramas que se han visto una vez (**singleton**) para calcular una estimación de los N -gramas que no aparecen ninguna vez en el *training set*, dividiendo el número de N -gramas que han aparecido una vez entre el número total de N -gramas del *training set*. El resultado de esta división es la probabilidad de los N -gramas que no han aparecido nunca.
- Good-Turing simple: similar al anterior pero se reemplazan los ceros antes de calcular el recuento suavizado.

4.4 Interpolación y Backoff

Además del problema tratado por el *smoothing* aparece otro cuando los datos del corpus de entrenamiento no son del todo fiables o no disponemos de ejemplos en el *training set*. Para solucionarlo se utilizan dos algoritmos:

- Interpolación: se combinan los N -gramas de distinto orden interpolando todos los modelos (unigramas, bigramas, trigramas, etc.).
- Backoff: si el N -grama que se necesita no aparece en el *training set*, se aproxima retrocediendo hacia el $(N-1)$ -grama, y se continúa retrocediendo hasta que alguno de los datos tenga información adecuada.

4.5 Análisis morfológico-léxico aplicado a motores de búsqueda textuales

El análisis morfológico-léxico es un proceso fundamental que debe preceder al diseño y desarrollo de los motores de búsqueda textuales. El conocimiento y estudio de la estructura interna de las palabras, de sus características y propiedades, así como el modo en que se derivan y flexionan, ayudan a realizar búsquedas más efectivas, ya que incrementan la exhaustividad de los motores de búsqueda [1].

Un motor de búsqueda está constituido básicamente por un crawler y un índice [3]. El primer componente es el que realiza las búsquedas, y en el índice se almacenan los términos ordenados según algún criterio o regla. Separar la raíz de la palabra (stem) de los afijos permite obtener mejores resultados en los motores de búsqueda por palabras clave. Todos los términos que tengan la misma raíz se almacenan como una sola entrada en el índice. Se utilizan varios algoritmos de stemming, en los que el stem obtenido no tiene por qué coincidir con la raíz morfológica de la palabra. La función de stemming se emplea tanto en el proceso de búsqueda como en la posterior clasificación e indexación de los resultados que se muestran, reduciendo así el tamaño del índice. Este hecho redonda en un mayor rendimiento ya que, al haber menos términos almacenados, se incrementa la velocidad del motor de búsqueda.

La división de un documento en párrafos y frases, tokenización, también aumenta la eficacia de los resultados en los motores de búsqueda textuales. Muchos de éstos constan de un programa especializado en analizar el texto para descomponerlo en frases. En consecuencia, es fundamental llevar a cabo un análisis del texto antes de realizar la búsqueda en los documentos. Este estudio debe contar, entre otras, de las siguientes fases:

- Reconocer el idioma en que está escrito, para así aplicar el análisis morfológico-léxico apropiado.
- Tokenizar el documento, dividiéndolo en párrafos, frases y palabras. Se deben identificar los signos ortográficos, espacios, etc.
- Eliminar las palabras vacías: artículos, preposiciones, pronombres, etc.
- Identificar *N*-gramas: grupo de palabras consecutivas que forman una secuencia y se almacenan como un solo término.
- Lematizar las palabras, separando los afijos de la raíz.

5. Etiquetado

La importancia de la clase gramatical, también llamada conjunto de etiquetas, radica en la cantidad de información que proporciona sobre una palabra y las que aparecen próximas a ella. También se utiliza en el proceso de lematización para la recuperación de información, ya que la clase gramatical de una palabra nos indica que afijos morfológicos puede llevar. La asignación automática de la clase gramatical es fundamental en el análisis, en los algoritmos de desambiguación, y en el análisis de textos para encontrar nombres, fechas u otras entidades.

5.1 Conjunto de etiquetas para el inglés

El más popular es el del corpus Brown, compuesto por 87 etiquetas. Este corpus fue etiquetado primero con *part-of-speech* utilizando el programa TAGGIT y después corregido manualmente. Además de este, que fue el pionero, se emplea también el conjunto de etiquetas de Penn Treebank y el de la Universidad de Lancaster, CLAWS, utilizado por el British National Corpus (BNC).

Algunas palabras pueden ser etiquetadas de forma distinta según su contexto. Decidir qué etiqueta asignarle a cada palabra requiere un amplio conocimiento de la sintaxis. Otro problema se presenta a la hora de etiquetar palabras que pueden modificar a un nombre. También puede haber dificultad en distinguir un verbo en participio de un adjetivo. Todos estos problemas dan lugar a ambigüedad gramatical.

El conjunto de etiquetas utilizado por Penn Treebank es menor que el del corpus Brown. Esto es debido a que en el primero las frases no sólo se etiquetan, sino que también se analizan. Por lo tanto, en la estructura de la frase se incluye información sintáctica útil. Pero en algunas situaciones este etiquetado no es suficientemente específico. De hecho, la ausencia de algunas etiquetas hace que, para algunas tareas de procesamiento del

lenguaje natural de alto nivel, el conjunto de etiquetas de Penn Treebank sea menos eficiente.

5.2 Etiquetado *Part-of-Speech* o POS-tagging

Es el proceso de asignación de una categoría gramatical u otro marcador de clase sintáctico a cada palabra de un corpus. Puesto que las etiquetas incluyen los signos de puntuación, el primer paso es tokenizar el texto, es decir, separar los puntos, comas, signos de admiración, etc., de las palabras. Así se podría desambiguar, por ejemplo, el punto ortográfico, indicando si se trata de un marcador de final de una frase o forma parte de una abreviatura. En el apartado anterior se han mencionado otras situaciones en las que también se produce ambigüedad a la hora de etiquetar una palabra. Este es el principal problema del POS-tagging, elegir la etiqueta adecuada según el contexto.

La mayoría de los algoritmos que existen para realizar la desambiguación se pueden clasificar en: basados en reglas y probabilísticos o estocásticos. A continuación se describe un ejemplo de cada uno de ellos.

5.3 POS-tagging basado en reglas

Los desambiguadores basados en reglas utilizan una gran base de datos que contiene reglas de desambiguación escritas a mano. En una primera etapa, utilizan un diccionario para asignar a cada palabra una lista de posibles categorías gramaticales. Posteriormente, utilizan reglas de desambiguación para reducir esta lista a una sola categoría gramatical para cada palabra.

Uno de los enfoques basados en reglas más completo es la gramática de restricciones, que es la que utiliza el etiquetador grammatical EngCG ENGTWOL. Las restricciones se aplican de forma negativa, eliminando las etiquetas que no son coherentes con el contexto. ENGTWOL tiene alrededor de 90000 entradas léxicas, contando una palabra con varias categorías gramaticales como entradas separadas, y desecharndo formas flexivas y muchas formas derivadas. Cada entrada se anota con un conjunto de características morfológicas y sintácticas. Algunas de estas características reflejan si se trata de un comparativo, superlativo, demostrativo, singular, femenino, genitivo, nominativo, etc.

5.4 POS-tagging estocástico

Los desambiguadores estocásticos resuelven la ambigüedad utilizando un corpus de entrenamiento para calcular la probabilidad de que una palabra tenga una etiqueta determinada según un contexto dado.

En este tipo de modelos se necesita gran cantidad de datos para el entrenamiento. Sólo así es posible conseguir que la estimación de la probabilidad sea correcta. Como consecuencia, si en una frase alguna de las palabras tiene ambigüedad gramatical, es decir, es posible asignarle más de una etiqueta, el problema es computacionalmente muy complejo. Esto es debido a que se necesita determinar tanto la probabilidad previa de la secuencia de etiquetas (para cada una de las posibles etiquetas), como la probabilidad de la frase completa.

El etiquetador HMM (Hidden Markov Model), para resolver este problema, realiza dos suposiciones para simplificar este cálculo. Primera, la probabilidad de aparición de una palabra depende sólo de su propia etiqueta, es decir, es independiente de las palabras que tiene alrededor y de las etiquetas que éstas tengan. Segunda, la probabilidad de aparición de una etiqueta depende sólo de la etiqueta previa y no de la secuencia completa de etiquetas. Por tanto sólo tiene en cuenta bigramas.

Combinando estas dos suposiciones se facilitan las operaciones, ya que sólo hay que considerar dos tipos de probabilidades: las de transición de etiquetas, que representan la probabilidad de una etiqueta dada la previa; y las de palabras, que representan la probabilidad de que una etiqueta determinada esté asociada con una palabra dada.

Existe una extensión para el algoritmo del HMM que calcula la probabilidad de una etiqueta según las dos etiquetas previas, es decir, considera trigramas. Pero presenta el problema de escasez de los datos del que se habló anteriormente, ya que puede haber un trígrama que ocurra en el *test set* pero que no aparezca en el *training set*. Esto implica la asignación de una probabilidad de 0 a este trígrama, que se resuelve igual que el apartado 4.3, con técnicas de suavizado e interpolando los resultados.

En el caso de que no se disponga de datos de entrenamiento, para idiomas que no han sido etiquetados previamente, se puede utilizar el algoritmo EM (Entropy Model).

5.5 Etiquetado basado en transformación (Transformation-Based tagging)

Este etiquetado, también llamado etiquetado de Brill o aproximación de Brill, se basa en los tipos de etiquetadores comentados anteriormente, basados en reglas y estocásticos. Es decir, para especificar qué etiqueta se debe asignar a cada palabra utiliza reglas que

son inducidas automáticamente a partir de los datos, utilizando técnicas de aprendizaje supervisado y un corpus de entrenamiento.

Este método consta de un conjunto de reglas de etiquetado. Se etiqueta un corpus según la regla más general. A continuación se elige una regla un poco más específica, que cambia algunas de las etiquetas asignadas. Posteriormente, se aplica una algo más restringida y se cambian las etiquetas afectadas. El algoritmo que utiliza tiene tres fases principales:

- Se marca cada palabra con la etiqueta más probable.
- Examina las posibles transformaciones que se le pueden aplicar y selecciona la que consigue una mejora superior en el etiquetado.
- Asigna esta nueva etiqueta a la palabra.

Se repiten los dos últimos pasos hasta que se alcanza un criterio de parada, como por ejemplo no haber logrado suficiente mejora en la pasada anterior. De esta manera, se establece un “procedimiento de etiquetado” que se puede aplicar a otro corpus. El número de transformaciones se debe limitar de alguna manera, ya que el algoritmo debe observar todas las modificaciones posibles para seleccionar la mejor. Para ello se utiliza un pequeño conjunto de plantillas o transformaciones abstractas:

- La palabra precedente (siguiente) está etiquetada como z .
- La palabra situada dos por delante (detrás) está etiquetada como z .
- Una de las dos palabras precedentes (siguientes) está etiquetada como z .
- Una de las tres palabras precedentes (siguientes) está etiquetada como z .
- La palabra precedente está etiquetada como z y la siguiente está etiquetada como w .
- La palabra precedente (siguiente) está etiquetada como z y la palabra situada dos por delante (detrás) está etiquetada como w .

Roche y Schabes [15] muestran como se puede aumentar la velocidad del etiquetador convirtiendo cada regla en un transductor de estado finito y realizando la composición de todos los transductores.

6. Conclusiones

El PLN estudia mecanismos para establecer una comunicación efectiva hombre-máquina. Para ello es fundamental el conocimiento del lenguaje. En este trabajo se profundiza en el aspecto morfológico-léxico. Entre los modelos más utilizados en PLN están los autómatas de estado finito, que se pueden especificar a través de una expresión

regular. Cada expresión regular tiene un autómata finito equivalente. Tanto las expresiones regulares como los autómatas finitos se pueden utilizar para determinar lenguajes regulares. Estos, a su vez, pueden ser generados por gramáticas regulares.

Los autómatas finitos pueden ser deterministas o no deterministas. En los primeros, cada estado de entrada define de forma precisa el estado de salida. En los segundos, se presentan puntos de decisión, en los que es posible seleccionar un camino erróneo. En este caso se debe volver al punto de decisión y valorar otra opción hasta que se llega a un estado de aceptación. Esto supone una búsqueda en el espacio de estados gracias a un mecanismo de agenda, que almacena los caminos que aún no han sido explorados.

Un transductor de estado finito es un autómata finito que enlaza dos conjuntos de símbolos, de tal forma que lee una cadena y genera otra. Los transductores secuenciales son un subconjunto de transductores que son deterministas en sus entradas, aunque pueden no serlo en su salida. Para el análisis morfológico que nos ocupa en este estudio, un transductor se asimila a un traductor que coge como entrada una secuencia de letras y produce como salida una cadena de morfemas. Las palabras se construyen a partir de morfemas, que se pueden dividir en raíz (stem) y afijos. Las formas más comunes de combinar morfemas para construir palabras son: inflexión, derivación, composición y uso de clíticos. En este trabajo se muestra un estudio básico de la morfología de los idiomas inglés y español.

El exhaustivo análisis morfológico-léxico de los documentos o textos es fundamental en PLN. Además, permite mejorar la exhaustividad y precisión de los motores de búsqueda. En este proceso se deben seguir varios pasos:

- Reconocer el idioma para establecer la morfología de sus palabras.
- Tokenizar el texto para dividirlo en párrafos, frases y palabras.
- Eliminar las palabras vacías.
- Identificar N -gramas.
- Lematizar las palabras para separar los afijos de la raíz.

Existen diversos algoritmos para lematizar tanto las consultas como las palabras de los documentos. Uno de los más utilizados es el de Porter, que utiliza las propiedades de composición e intersección de transductores para construir un lematizador.

Por su parte, el modelo de N -gramas es una de las herramientas más útiles en el análisis de textos. Se trata de un modelo probabilístico que utiliza datos estadísticos para predecir la siguiente palabra en una secuencia. Para calcular las probabilidades y

comparar los resultados en estos modelos, se divide el corpus de documentos en un grupo de entrenamiento (*training set*) y otro de pruebas (*test set*). El *training set* debe ser suficientemente grande y el *test set* adecuadamente pequeño. Cuando se presentan problemas de escasez de datos en el *training set*, se utilizan técnicas como el suavizado o la interpolación para resolverlos. El mejor modelo será aquel que asigne una probabilidad más alta a los datos del test y un valor de perplejidad más bajo. Esto se debe a que la perplejidad es inversa a la probabilidad. La principal ventaja de los modelos probabilísticos es su capacidad para resolver problemas de ambigüedad.

El modelo de *N*-gramas se utiliza en varios campos de investigación, uno de los cuales es el etiquetado gramatical. La importancia del etiquetado radica en la información que proporciona sobre una palabra y sus adyacentes, lo que permite su utilización en algoritmos de desambiguación. Se analizan en este trabajo los principales tipos de etiquetado que existen y como resuelven los problemas de ambigüedad.

PARTE II. GRAMÁTICAS DE CONTEXTO LIBRE, PARSING, UNIFICACIÓN DE RASGOS Y SEMÁNTICA Y ANÁLISIS SEMÁNTICO

Resumen. Las gramáticas de contexto libre constan de un conjunto de reglas con las que se pueden generar expresiones de un lenguaje y asignar una estructura a una frase determinada. Estas reglas se expanden para representar un árbol de análisis. El análisis sintáctico consiste en buscar el árbol correcto para cada frase según su contexto. Cuando la gramática asigna más de un árbol de análisis a una misma frase se produce la ambigüedad estructural, que se resuelve con métodos de programación dinámica. Por su parte, las gramáticas de unificación representan la estructura sintáctica de la lengua mediante gramáticas de contexto libre aumentadas con extensiones para incrementar su expresividad (rasgos). La unificación es un proceso que combina la información de dos o más estructuras de rasgos compatibles para obtener una nueva que contenga la información de todas ellas. En el estudio del lenguaje natural, además del análisis sintáctico, se debe analizar el sentido, interpretación y significado de las palabras. La semántica de una palabra o frase puede capturarse mediante estructuras formales que cumplan una serie de características: verificables, sin ambigüedad, precisas y expresivas. La lógica de primer orden se utiliza para representar este conocimiento y la representación del significado, y permite la inferencia o deducción de nuevas proposiciones válidas para el dominio que se está modelando. La representación del significado consta de estructuras compuestas que se corresponden con los objetos o palabras del dominio, y analiza también sus propiedades y relaciones (polisemia, hominimia, hiponimia, sinonimia, etc.).

7. Introducción

La gramática estudia las reglas y principios que rigen el uso de la lengua y la organización de las palabras dentro de las frases y otros constituyentes sintácticos. El análisis de la lengua a nivel sintáctico abarca las relaciones de concordancia entre las palabras, que establece la forma en la que se colocan unas al lado de otras.

Una gramática formal es un conjunto de reglas para escribir cadenas de caracteres, junto con un símbolo inicial desde el cual debe comenzar. Las gramáticas generativas son el tipo más conocido de gramáticas formales. A su vez, existen cuatro tipos diferentes de gramáticas generativas, definidas por la clase de reglas que contienen. La *jerarquía de Chomsky* clasifica estos tipos según su poder generativo: irrestrictas, dependientes del contexto, libres de contexto y regulares o de estados finitos, ordenadas de mayor a menor poder generativo. Cada tipo de gramática genera o define un tipo de lengua y se asocia con un tipo de autómata (mecanismo abstracto que realiza operaciones según un conjunto de instrucciones iniciales).

Las gramáticas de contexto libre (CFG, Context-Free Grammar) modelan la estructura constituyente del lenguaje natural. Se pueden utilizar tanto para generar frases como para asignar una estructura a una frase. Constan de un conjunto de reglas que se expanden para representar un árbol de análisis. El análisis sintáctico consiste en buscar todos los posibles árboles, seleccionando el correcto para cada frase según su contexto.

Cuando la gramática asigna más de un árbol de análisis a una misma frase se produce la ambigüedad estructural. Para resolverla se requieren conocimientos estadísticos, semánticos y pragmáticos. Los métodos de programación dinámica (algoritmo CKY, Early y análisis gráfico) se emplean para solucionar dichos problemas de ambigüedad.

Por su parte, las gramáticas de unificación representan la estructura sintáctica de la lengua mediante gramáticas sintagmáticas de contexto libre, aumentadas con extensiones para incrementar la expresividad de los formalismos, como por ejemplo la utilización de rasgos. Por lo tanto, en las gramáticas de contexto libre, los elementos fundamentales son las categorías gramaticales, mientras que en las de unificación son los rasgos (segmentos de información lingüística por debajo de la categoría). La unificación es un proceso que combina la información de dos o más estructuras de rasgos, que deben ser compatibles, para obtener una nueva estructura de rasgos que contenga toda la información de las estructuras originales.

En el estudio del lenguaje natural, además del análisis sintáctico, se debe analizar el sentido, interpretación y significado de las palabras. La semántica de una palabra o frase puede capturarse mediante estructuras formales como la lógica de primer orden, que permite la inferencia o deducción de nuevas proposiciones válidas para el dominio que se está modelando. La representación del significado consta de estructuras compuestas que se corresponden con los objetos del dominio, así como con sus propiedades y relaciones (polisemia, hominimia, hiponimia, sinonimia, etc.)

A continuación, en la sección 8, se detallan diversos conceptos sobre las CFGs, la expansión de reglas y las relaciones sintagmáticas, de concordancia y dependencia que se establecen entre los constituyentes. La sección 9 se centra en el análisis sintáctico como un problema de búsqueda del árbol de análisis correcto para una frase según su contexto. En la sección 10 se define la gramática de unificación de rasgos y sus componentes. Finalmente, en la sección 11 se analiza el sentido, significado e interpretación de las palabras y frases del lenguaje natural. Como anexo aparece un **Glosario de términos**.

8. Gramáticas de contexto libre para el análisis del lenguaje natural

Una CFG, llamada también gramática de estructura sintagmática o de frase, modela la estructura constituyente del lenguaje natural, y fue formalizada por Chomsky y Backus. Las CFGs se pueden utilizar tanto para generar frases, por lo que se llaman gramáticas generativas, como para asignar una estructura a una frase determinada. Esto se realiza con una secuencia de expansión de reglas, llamada derivación de cadenas de palabras, que se representa con un árbol de análisis (o con notación entre corchetes).

El lenguaje formal definido por una CFG es el conjunto de cadenas que se pueden derivar de un **símbolo inicial**. Cada gramática debe tener un símbolo inicial

determinado, llamado S , que representa un nodo “sentencia”, y el conjunto de cadenas generadas a partir de S es el conjunto de sentencias de alguna versión simplificada del idioma.

8.1 Definición formal de CFG

Se define por la tupla:

- N conjunto de símbolos no terminales (o variables)
- Σ conjunto de símbolos terminales
- R conjunto de reglas o producciones, de la forma $A \rightarrow \beta$, donde A es un no terminal, y β es una cadena de símbolos del conjunto de cadenas $(\Sigma \cup N)^*$ (terminal o no terminal)
- S un símbolo inicial determinado

Si $A \rightarrow \beta$ es una regla de producción de P y α y γ son cadenas del conjunto $(\Sigma \cup N)^*$, se dice que $\alpha A \gamma$ deriva directamente $\alpha \beta \gamma$, o $\alpha A \gamma \Rightarrow \alpha \beta \gamma$

En consecuencia, se puede definir formalmente el lenguaje L_G generado por la gramática G como el conjunto de cadenas compuesto por símbolos terminales que pueden ser derivados a partir del símbolo inicial S .

$$L_G = \{w \mid w \text{ está en } \Sigma^* \text{ y } S \xrightarrow{*} w\}$$

El análisis sintáctico consiste en transformar una cadena de palabras en su árbol de análisis sintáctico correspondiente. Existen varios algoritmos para realizar este análisis, detallados en la sección 3 de este trabajo.

8.2 Reglas gramaticales

8.2.1 Construcciones a nivel de frase

Existen muchos tipos de construcciones, pero los más importantes son tres: declarativas, imperativas e interrogativas. Estas últimas se pueden dividir en totales (preguntas de sí o no) y parciales (piden información).

	Español	Inglés
Declarativas	La sopa está sobre la mesa	The editors have not yet published his novel
Imperativas	Dame ese libro	Tell me the truth!
Interrogativas s/n	¿Compró Juan algún libro?	Do you walk to work every day?
Interrogativas wh	¿Dónde vas de vacaciones?	When will he return?

8.2.2 Los sintagmas

Un sintagma es una unidad gramatical formada por palabras que tienen una función sintáctica determinada. Cada sintagma tiene un núcleo que aporta las características básicas al sintagma. Hay diversos tipos:

- Nominal (SN): su núcleo es un nombre o pronombre. Delante del nombre puede haber post-determinantes: números cardinales (two, dos), números ordinales (first, next, primero, siguiente) y cuantificadores (many, muchos). Por lo tanto, combinando las reglas se obtiene:

$$SN \rightarrow (Det.) (Card.) (Ord.) (Cuant.) (Frase Adjetiva) Nominal$$

Detrás del nombre puede haber post-modificadores y antes del sintagma nominal puede haber pre-determinantes.

- Verbal (SV): su núcleo es un verbo, y puede incluir algunos complementos. Un complemento del SV puede ser otro SV. Los verbos pueden llevar partículas para formar verbos compuestos o frasales. En este caso, la partícula se considera una parte integral del verbo. En general, los verbos se pueden categorizar en transitivos (llevan un objeto directo) e intransitivos (no llevan objeto directo). La relación entre los verbos y sus complementos en una CFG se puede representar por las siguientes reglas:

$$SV \rightarrow Verbo$$

$$SV \rightarrow Verbo + SN$$

$$SV \rightarrow Verbo + S$$

Los verbos auxiliares imponen restricciones sintácticas particulares al verbo siguiente. Incluyen los verbos modales (can, could, may, must, would, etc., poder, deber, querer, soler, etc.), el auxiliar perfecto *have/haber*, el auxiliar progresivo *be/ser* y el auxiliar pasivo *be*.

- Preposicional, adjetival y adverbial: sus núcleos son una preposición, un adjetivo y un adverbio, respectivamente.

8.2.3 Concordancia

Cada uno de los constituyentes de la gramática debe concordar con aquellos otros elementos con los que se relaciona. Existen concordancias de género, número, persona, tiempo y modo. Al añadir las reglas para incluir todas las posibles concordancias el tamaño de la gramática se multiplica. Para evitarlo se puede utilizar la parametrización de cada no terminal con estructuras de rasgos y unificación.

8.2.4 Coordinación

Todas las frases se pueden unir por medio de conjunciones (and, or, but, y, o, pero) para formar una frase mayor. Las reglas para la coordinación son:

$$SN \rightarrow SN \text{ conj } SN$$

$$SV \rightarrow SV \text{ conj } SV$$

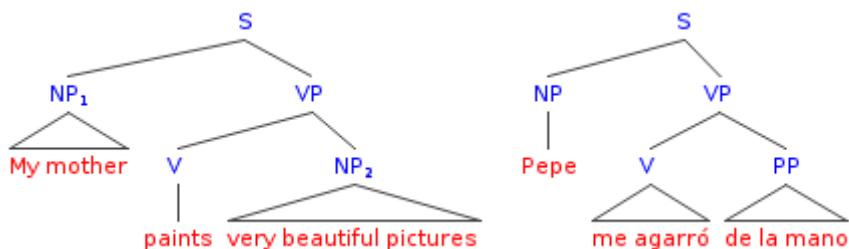
$$S \rightarrow S \text{ conj } S$$

8.3 Treebanks

Las reglas de las CFGs vistas hasta ahora se pueden utilizar para asignar un árbol de análisis a cada frase. Esto significa que se puede construir un corpus en el que cada frase está anotada sintácticamente con un árbol de análisis. El corpus anotado de esta manera se llama *treebank*. Existen *treebanks* sobre diversos temas y para distintos idiomas. Normalmente, son analizados automáticamente y corregidos posteriormente a mano por un lingüista.

Las búsquedas en *treebanks* se utilizan para encontrar ejemplos de fenómenos gramaticales especiales, para investigaciones lingüísticas y para responder preguntas sobre una aplicación computacional. Además de las expresiones regulares y booleanas, se necesita un lenguaje que pueda especificar restricciones sobre nodos y enlaces en un árbol de análisis, así como para buscar patrones específicos.

Ejemplos de árboles sintácticos son los siguientes:



8.4 Equivalencia gramatical y forma normal

Se define un lenguaje formal como un conjunto, posiblemente infinito, de cadenas de palabras. Dos CFGs son equivalentes si generan el mismo conjunto de cadenas. Hay dos tipos de equivalencia en gramática: débil y fuerte. Dos gramáticas son fuertemente equivalentes si generan el mismo conjunto de cadenas y asignan la misma estructura de frase a cada oración. Son débilmente equivalentes si generan el mismo conjunto de cadenas pero no asignan la misma estructura de frase a cada oración.

Cualquier CFG se puede convertir en una gramática en *forma normal de Chomsky* (CNF) débilmente equivalente si no tiene símbolos ϵ y, además, cada producción es de la forma $A \rightarrow B C$ o $A \rightarrow a$. Es decir, la parte derecha de cada regla tiene dos símbolos no terminales o un solo símbolo terminal. Por lo tanto, si la gramática está en CNF, los árboles de análisis son binarios:

$$\begin{aligned} A \rightarrow B C D & \implies A \rightarrow B X \\ & \quad X \rightarrow C D \end{aligned}$$

De esta forma se pueden producir gramáticas más pequeñas, ya que se podría reducir el número de reglas. La generación de un símbolo A con una secuencia potencialmente infinita de símbolos B con una regla de la forma $A \rightarrow A B$ se conoce como la *adición de Chomsky*.

8.5 Estado finito y gramáticas de contexto libre

Los modelos gramaticales deben ser capaces de representar hechos complejos, subcategorización y relaciones de dependencia sobre constituyentes que están relacionados entre sí. Para ello se necesita la potencia de las CFGs. Si se pudieran utilizar métodos de estado finito, la velocidad de procesamiento sería mayor. Pero hay dos razones para utilizar CFGs en lugar de lenguajes regulares:

- Determinadas estructuras sintácticas hacen que los lenguajes naturales no sean regulares.
- Los métodos de estado finito no expresan de forma sencilla la generalización, que es fundamental para la realización de formalismos comprensibles que produzcan estructuras reutilizables.

Las reglas en una gramática regular son una forma restringida de las reglas de una CFG. En una gramática lineal por la derecha, los no terminales se expanden a una cadena de terminales o a una cadena de terminales seguida por un no terminal ($A \rightarrow w^* A \rightarrow w^*$). Son similares a las reglas de contexto libre, pero no pueden expresar reglas que son recursivas por el centro, donde un no terminal se reescribe como a sí mismo, rodeado por cadenas: $A \xrightarrow{*} \alpha A \beta$.

Es decir, un lenguaje puede ser generado por una máquina de estado finito si, y solo si, la gramática que genera L no tiene recursiones por el centro. Esto es debido a que las reglas gramaticales en las que los símbolos no terminales están siempre a la derecha o a la izquierda de una regla se pueden procesar iterativamente, pero no recursivamente.

Gran parte de las reglas se pueden utilizar con métodos finitos. De hecho, es posible construir automáticamente una gramática regular que sea una aproximación a una CFG. Por lo tanto, para muchos propósitos prácticos en los que no es necesario que coincidan las reglas sintácticas y semánticas, las reglas de estado finito son suficientes.

8.6 Gramáticas de dependencia

Muchos *treebanks* y analizadores sintácticos disponibles utilizan como representación CFGs. Pero existen unos formalismos gramaticales, llamados *gramáticas de dependencia*, que están adquiriendo gran importancia en el procesamiento del lenguaje. En este tipo de gramáticas no sólo se consideran los elementos léxicos que integran la estructura sintáctica, sino también las relaciones, tanto semánticas como sintácticas, existentes entre ellos.

Una de las ventajas de los formalismos de dependencias es que permiten el análisis predictivo de la relación que puede existir entre una palabra y las que dependen de ella. Por ejemplo, conocer el verbo puede ayudar a decidir qué sustantivo es el sujeto o cuál es el objeto.

Las implementaciones computacionales de las gramáticas de dependencia incluyen la gramática de enlaces (*Link Grammar*), la gramática de restricciones (*Constraint Grammar*), MINIPAR, DepPattern, y el Analizador de Stanford.

9. Análisis sintáctico

Es el proceso de reconocer una cadena de entrada y asignarle una estructura sintáctica. Para poder realizarlo es necesario especificar algoritmos que utilicen las gramáticas de contexto libre para producir árboles sintácticos. Se parte del símbolo inicial *S*, que se puede considerar como un nodo del árbol de análisis, por debajo del cual el verbo principal de *S* tiene todos sus argumentos

9.1 El análisis como búsqueda

Un analizador sintáctico explora todos los posibles árboles de análisis, con el objetivo de encontrar el árbol sintáctico correcto para una frase determinada. La forma más simple de describir cómo en una gramática se puede derivar una cadena es listar las cadenas de símbolos consecutivas, comenzando por el símbolo inicial *S* y finalizando con la cadena y las reglas que se han aplicado para obtenerla. Las dos estrategias de búsqueda que se utilizan en la mayoría de analizadores se detallan a continuación.

9.1.1 Análisis top-down

Este tipo de análisis construye el árbol desde el nodo raíz hacia las hojas. Comenzando en S , el algoritmo busca todas las reglas gramaticales que tienen a S en el lado izquierdo y construye sus árboles. El siguiente paso es expandir los constituyentes de esos nuevos árboles. En cada nivel o capa del espacio de búsqueda se utiliza el lado derecho de las reglas para proporcionar nuevos conjuntos de posibles constituyentes, que se utilizan recursivamente para generar el resto de árboles. Los árboles crecen hacia abajo hasta que se llega a las categorías gramaticales que están en la parte inferior. Los árboles cuyas hojas no se adapten a todas las palabras de la entrada se rechazan, dejando sólo los árboles que representan un análisis correcto.

9.1.2 Análisis bottom-up

El analizador comienza con las palabras de la entrada, y construye árboles a partir de ellas hacia arriba, aplicando una regla cada vez. El análisis tiene éxito si el analizador puede construir un árbol cuya raíz es el símbolo inicial S y contiene todas las palabras de la entrada. El análisis comienza construyendo árboles parciales con la categoría gramatical de cada palabra. Si una palabra es ambigua, el analizador debe considerar todas las etiquetas gramaticales que se le puedan asignar.

Cada árbol del segundo nivel se expande buscando lugares en el proceso de análisis dónde podría encajar la parte derecha de una regla, al contrario que en el *top-down*, que expande los árboles aplicando reglas cuando sus partes izquierdas encajan con un no terminal que aún no se ha expandido.

9.1.3 Comparación de los análisis top-down y bottom-up

Cada uno tiene sus ventajas e inconvenientes. La estrategia *top-down* no pierde tiempo explorando árboles que no pueden dar como resultado S . Por el contrario, en la estrategia *bottom-up* se generan gran cantidad de árboles que no pueden conducir al símbolo S ni encajar con ningún árbol cercano.

Por otra parte, el enfoque *top-down* emplea mucho esfuerzo con árboles que no son consistentes con la entrada, puesto que genera árboles antes de examinarla. Sin embargo, el análisis *bottom-up*, no sugiere árboles que no se basan, al menos localmente, en la entrada.

9.2 Ambigüedad

En las estructuras sintácticas que se utilizan en el análisis se plantea un nuevo tipo de ambigüedad, la estructural. Esta aparece cuando la gramática asigna más de un posible análisis a una frase. Las dos formas más comunes de este tipo de ambigüedad son la adjunta y la de coordinación.

Una frase tiene ambigüedad adjunta si uno de sus constituyentes se puede adjuntar al árbol de análisis sintáctico en más de un lugar. En la ambigüedad de coordinación, se pueden unir distintos conjuntos de frases mediante una conjunción como *y*.

La mayoría de los sistemas de procesamiento del lenguaje natural deben ser capaces de elegir el análisis correcto de entre todos los posibles mediante un proceso conocido como *desambiguación sintáctica*. Para que sea efectivo, este proceso requiere conocimientos sobre estadística, semántica y pragmática que no están disponibles durante el procesamiento sintáctico. Por lo tanto, se deberían devolver todos los posibles árboles de análisis para una entrada determinada. Esto es difícil de llevar a cabo, debido al número exponencial de análisis que se pueden realizar para ciertas entradas.

Incluso si una frase no es ambigua en su conjunto, el análisis sintáctico puede no ser eficiente debido a la *ambigüedad local*, que ocurre cuando una parte de la frase es ambigua.

9.3 Buscar en la superficie de la ambigüedad

Explorar todos los posibles árboles de análisis en paralelo requeriría una cantidad de memoria inimaginable para almacenarlos. Una posible alternativa a la exploración de espacios de búsqueda complejos es utilizar una estrategia de retroceso basada en agenda, similar a la que se utiliza en los transductores. Este enfoque de retroceso expande el espacio de búsqueda incrementalmente, a través de la exploración de un estado cada vez. La elección del estado a expandir se puede hacer con métodos como primero en profundidad o primero en anchura, o con métodos más complejos que utilizan procedimientos probabilísticos o semánticos. Al llegar a un árbol inconsistente con la entrada, la búsqueda continúa volviendo a las opciones no exploradas que hay en la agenda.

La ambigüedad generalizada en las gramáticas lleva a una gran ineficiencia de los enfoques de retroceso. Los analizadores de retroceso construyen árboles parciales para la entrada y después los descartan para, posteriormente, tener que volver a construirlos.

9.4 Métodos de análisis de programación dinámica

Estos métodos resuelven el problema de la ambigüedad gracias a los algoritmos de *Minimum Edit Distance*, *Viterbi* y *Forward*. Funcionan rellenando de forma sistemática las tablas de solución de los sub-árboles para cada constituyente de la entrada. Su eficiencia radica en que estos sub-árboles se descubren una sola vez, se almacenan, y después se utilizan en todos los análisis en los que aparezca ese constituyente. Los tres métodos más utilizados son el algoritmo *Cocke-Kasami-Younger* (CKY), el algoritmo de *Earley* y el *análisis gráfico*.

9.4.1 Análisis CKY

La gramática utilizada con este algoritmo debe estar en la *forma normal de Chomsky* (CNF), especificada en el punto 2.4. La parte derecha de cada regla debe expandir a dos no terminales o a un solo terminal. El proceso de conversión completo se puede resumir de la siguiente forma:

1. Copiar todas las reglas correctas a la nueva gramática sin cambiarlas.
2. Convertir los terminales que aparecen dentro de las reglas a no terminales vacíos.
3. Convertir las producciones unitarias.
4. Hacer que todas las reglas sean binarias y añadirlas a la nueva gramática.

De esta manera, cada nodo no terminal tiene exactamente dos hijos. En consecuencia, para codificar la estructura completa de un árbol de análisis se puede utilizar una matriz o tabla bi-dimensional, donde cada constituyente se representa por una entrada $[i, j]$ en la tabla. Para llenar correctamente esta tabla se comienza de abajo hacia arriba, de tal forma que cuando se está introduciendo la celda $[i, j]$, las celdas de su izquierda y las que están por debajo de ella (contienen partes que podrían contribuir en su entrada) ya se han llenado.

Como el número de análisis que se pueden asociar a una entrada crece exponencialmente, se suelen añadir a la tabla las probabilidades para cada entrada, lo que hace posible elegir el árbol de análisis más viable. Pero la conversión a la forma normal de Chomsky complicará cualquier enfoque dirigido por la sintaxis para realizar el análisis semántico. Para tratar de resolver este problema se debe mantener suficiente información sobre la transformación realizada en los árboles de análisis. Esto permitirá poder regresar a la gramática original después del análisis.

9.4.2 El algoritmo de Earley

Utiliza la programación dinámica para implementar una búsqueda desde arriba hacia abajo. Va desde la izquierda hacia la derecha llenando un array o tabla que tiene $N + 1$ entradas. Para cada posición de una palabra en la frase, la tabla contiene una lista de estados que representan los árboles de análisis parciales que han sido generados hasta el momento. Al final de la frase, la tabla codifica de forma compacta todos los posibles análisis de la entrada. Cada sub-árbol se representa sólo una vez y se comparte por todos los análisis que lo necesiten.

Los estados individuales contenidos en cada entrada de la tabla tienen tres tipos de datos: un sub-árbol que se corresponde con una regla de la gramática, información relativa al progreso realizado en la construcción de este sub-árbol, y la posición del sub-árbol con respecto a la entrada.

Este algoritmo recorre los $N + 1$ conjuntos de estados de la tabla de izquierda a derecha, procesando los estados en orden. En cada paso, se aplica uno de los tres operadores (PREDICTOR, SCANNER o COMPLETER) a cada estado, dependiendo de su posición. Esto provoca la creación de nuevos estados o pasar al siguiente conjunto de estados de la tabla. Los estados nunca se eliminan, y el algoritmo nunca retrocede hacia una entrada previa. La presencia del estado $S \rightarrow \alpha \bullet, [0, N]$ en la última entrada de la tabla indica un análisis con éxito. El símbolo \bullet indica en qué punto está el análisis.

Estos dos algoritmos, CKY y Earley, no son analizadores, sino reconocedores. En ambos casos, para poder crear el árbol sintáctico a partir de los resultados obtenidos después de su aplicación, se necesitan campos adicionales que suministren la información necesaria para completar el estado generado por cada constituyente.

9.4.3 Análisis gráfico

Los dos algoritmos anteriores son estáticos. En cambio, el análisis gráfico permite una determinación dinámica del orden en el que se procesan las entradas. Esto se realiza gracias al esquema de *agenda*, en el que cada estado que se crea se añade a una agenda que se mantiene en orden, según una política especificada aparte del algoritmo de análisis.

Los estados se procesan siguiendo la *regla fundamental*. Esta regla establece que cuando el gráfico contiene dos estados contiguos y uno de ellos contiene un constituyente que el otro necesita, se crea un nuevo estado que expande a los originales e incorpora el material proporcionado. Esta regla fundamental es una generalización de

las operaciones de relleno de la tabla de los algoritmos CKY y Earley. Si tenemos $A \rightarrow \alpha B \bullet \beta$, [i,j] y $B \rightarrow \gamma \bullet$, [j,k], entonces se añade el nuevo estado $A \rightarrow \alpha B \bullet \beta$ [i,k]. Este algoritmo selecciona un estado, y si está completo examina el gráfico para ver si alguno de los anteriores se puede avanzar. Si está incompleto, mira a los posteriores para ver si puede ser avanzado por alguno de los estados que ya existen en el gráfico. Posteriormente se realizan predicciones teniendo en cuenta los eventos que desencadenan dichas predicciones y la naturaleza de las mismas. Si la predicción es *top-down*, se desencadenan por las expectativas que surgen de los estados incompletos introducidos en la tabla. Si es *bottom-up*, se desencadenan por el descubrimiento de constituyentes completados.

9.5 Análisis parcial

Algunas tareas de procesamiento del lenguaje no requieren un árbol de análisis completo para todas sus entradas. Entonces se utiliza el análisis parcial o superficial. Algunos de los enfoques utilizados para este tipo de análisis emplean transductores en cascada, que producen árboles más simples porque aplazan las decisiones que requieren factores semánticos o contextuales.

Un enfoque alternativo se conoce como *chunking* o fragmentación. Este proceso identifica y clasifica las partes de una frase que no se solapan y que constituyen las frases gramaticales básicas de la mayoría de las gramáticas de cobertura amplia. Este conjunto de frases incluye las partes del discurso que mantienen el contenido. Puesto que los textos fragmentados no tienen una estructura jerárquica, es suficiente una notación con corchetes simples para denominar la localización y el tipo de fragmentos de un ejemplo dado. No es necesario que todas las palabras de la frase tengan asignada su etiqueta gramatical. Existen distintos tipos de fragmentación: basada en reglas de estado finito y basada en aprendizaje automático supervisado.

Para evaluar los sistemas de fragmentación se compara la salida dada por el fragmentador con las respuestas estándar proporcionadas por los anotadores humanos. Pero en lugar de hacerlo palabra por palabra, se evalúan según la precisión, exhaustividad y una medida que se utiliza en recuperación de la información.

$$\text{Precisión} := \frac{\text{Número de fragmentos correctos dados por el sistema}}{\text{Número total de fragmentos dados por el sistema}}$$

$$\text{Exhaustividad} := \frac{\text{Número de fragmentos correctos dados por el sistema}}{\text{Número total de fragmentos actuales que hay en el texto}}$$

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2P + R}$$

El parámetro β mide diferencialmente la importancia de la exhaustividad y la precisión, basándose en las necesidades de la aplicación. Un valor de β mayor que 1 favorece la exhaustividad, mientras que valores de β menores que 1 benefician la precisión. Si $\beta = 1$, las dos medidas están balanceadas.

10. Unificación de rasgos

Es necesario imponer una serie de restricciones a las reglas independientes del contexto puesto que en su estado más simple permiten combinaciones agramaticales. Los formalismos basados en estas restricciones representan información sobre la concordancia en número y persona, la subcategorización y las categorías semánticas. Además, modelan fenómenos más complejos que los que pueden realizarse con las gramáticas de contexto libre, pueden calcular eficientemente y de forma conveniente la semántica para las representaciones sintácticas y solucionan algunos de los problemas de sobre-generación de etiquetas.

10.1 Estructuras de rasgos

Un rasgo es un par atributo-valor, donde un atributo es un símbolo (átomo o elemento constante sin estructura interna), que da nombre al rasgo, y un valor es o bien un símbolo o bien una estructura de rasgos (símbolo complejo con una estructura jerárquica interna). Estas estructuras de rasgos se representan en una *matriz atributo-valor* o AVM. Por ejemplo, un sintagma nominal cuya categoría gramatical es 3sgNP, se puede representar con la siguiente matriz:

<i>CAT</i>	<i>SN</i>
<i>NUMERO</i>	<i>sg</i>
<i>PERSONA</i>	<i>3ra</i>

Existen estructuras de rasgos que contienen características compartidas, llamadas *estructuras de reentrada*. Las *estructuras de reentrada* se representan en la AVM añadiendo un índice numérico que indica los valores que comparten. Esto permite representar generalizaciones lingüísticas de una forma muy compacta y elegante. Supongamos que definimos un sintagma nominal y queremos establecer que la concordancia entre el núcleo nominal, N, y sus especificadores, ESP (artículos,

cuantificadores, etc.) comparten la misma estructura, representada por el rasgo complejo *concordancia*:

CAT	SN
N	<i>concordancia</i> 1
ESP	[<i>concordancia</i> 1]

	<i>NÚMERO sing</i>
	<i>GÉNERO fem</i>
	<i>PERSONA 3ra</i>

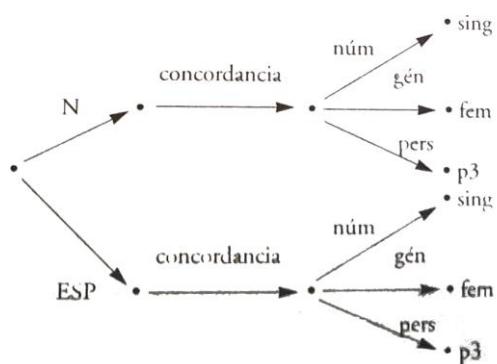
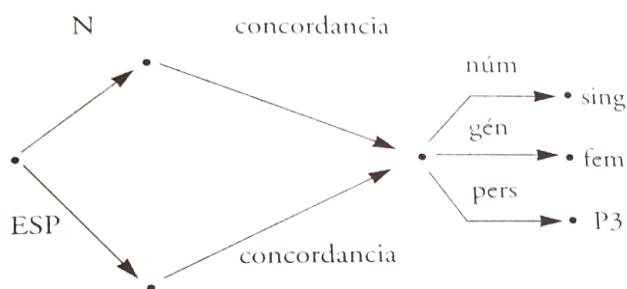
El recuadro con el número 1 representa la estructura compartida, que se especifica una sola vez, y en la segunda sólo se escribe el cuadro indexado. Es fundamental diferenciar una estructura compartida de una estructura con valores similares:

CAT	SN
N	<i>concordancia</i>
ESP	[<i>concordancia</i>]

	<i>NÚMERO sing</i>
	<i>GÉNERO fem</i>
	<i>PERSONA 3ra</i>

	<i>NÚMERO sing</i>
	<i>GÉNERO fem</i>
	<i>PERSONA 3ra</i>

Podría parecer que esta estructura y la anterior son equivalentes, pero no es así. La segunda es una estructura con rasgos que tienen valores semejantes, pero no asociados entre sí. Con grafos dirigidos se ve claramente la diferencia:



10.2 Unificación de estructuras de rasgos

La unificación es un proceso que combina la información de dos o más estructuras de rasgos para obtener una nueva estructura de rasgos más específica (tiene más información) que contenga toda la información dichas estructuras. La unificación de dos estructuras se rechaza si tienen características incompatibles. Por lo tanto, es una manera de fusionar la información de cada estructura de rasgos o de describir los objetos que satisfacen ambos conjuntos de restricciones.

$$[NÚMERO \ sing] \cup [PERSONA \ 3ra] = \begin{bmatrix} NÚMERO & sing \\ PERSONA & 3ra \end{bmatrix}$$

Una estructura de rasgos menos específica (más abstracta) *engloba* a una igual o más específica. Toda estructura de rasgos es englobada por la estructura vacía $[]$. La idea de subsunción permite la aplicación recursiva en la operación de unificación de las estructuras. La subsunción es una relación entre dos estructuras A y B, tal que si A subsume B ($A \sqsubseteq B$), entonces A es más general que B, o B es un ejemplo de A. Una estructura más general contiene menos información específica, por lo que será compatible con más objetos lingüísticos: $[NÚMERO \ sing] \sqsubseteq \begin{bmatrix} NÚMERO & sing \\ PERSONA & 3ra \end{bmatrix}$

La operación de unificación es *monotónica*, es decir, si alguna descripción es cierta en una estructura de rasgos, su unificación con otra produce una estructura de rasgos que también satisface la descripción original. Por lo tanto, también es asociativa, ya que dado un conjunto finito de estructuras de rasgos a las que se aplica la unificación, se puede realizar en cualquier orden, obteniendo el mismo resultado. La unificación es una forma de implementar la integración de conocimiento a partir de distintas restricciones, siempre que sean estructuras compatibles.

10.3 Estructuras de rasgos en gramática

Con la integración de las estructuras de rasgos y las operaciones de unificación en la especificación de una gramática se logran los siguientes objetivos:

- Asociar estructuras de rasgos complejas con elementos léxicos y con instancias de categorías gramaticales.
- Orientar la composición de estructuras de rasgos hacia constituyentes gramaticales más grandes, en base a las estructuras de rasgos de sus componentes.

- Aplicar restricciones de compatibilidad entre las partes especificadas de las construcciones gramaticales.

Utilizando el sistema PATR-II [17], la notación de la gramática para lograr estos objetivos es:

$$\beta_0 \rightarrow \beta_1 \dots \beta_n$$

{conjunto de restricciones}

dónde las restricciones tienen una de las siguientes formas:

$$\langle \beta_i \text{ camino del rasgo} \rangle = \text{valor atómico}$$

$$\langle \beta_i \text{ camino del rasgo} \rangle = \langle \beta_j \text{ camino del rasgo} \rangle$$

El *camino del rasgo* indica la secuencia de arcos que se pueden seguir desde el nodo raíz hasta la estructura de rasgos asociada con el componente β_i . Así, la restricción de la regla $S \rightarrow NP VP$ sólo si el número de NP es igual al de VP, se puede expresar de la siguiente manera:

$$\langle NP \text{ NUMERO} \rangle = \langle VP \text{ NUMERO} \rangle$$

En esta regla gramatical, se concatena un *NP* y un *VP* para formar una frase *S*. Con el nuevo esquema definido, esta concatenación debe estar acompañada por una operación de unificación correcta. Esto implica que se debe tener en cuenta la complejidad computacional de la operación de unificación y su efecto en la potencia generativa de la nueva gramática.

Se estudian a continuación las restricciones de unificación para algunos fenómenos lingüísticos: concordancia, subcategorización y dependencias de larga distancia.

10.3.1 Concordancia

La concordancia entre el sujeto y el verbo se puede expresar de la siguiente manera:

$$S \rightarrow NP VP$$

$$\langle NP \text{ CONCORDANCIA} \rangle = \langle VP \text{ CONCORDANCIA} \rangle$$

La concordancia entre determinantes y nombres en el sintagma nominal se indica así:

$$NP \rightarrow Det \text{ Nominal}$$

$$\langle Det \text{ CONCORDANCIA} \rangle = \langle Nominal \text{ CONCORDANCIA} \rangle$$

$\langle NP \text{ CONCORDANCIA} \rangle = \langle \text{Nominal CONCORDANCIA} \rangle$

10.3.2 Subcategorización

Cuando los verbos o sintagmas verbales no se pueden emparejar, hay que dividir la categoría del verbo en varias subcategorías. Pero este enfoque provoca que el número de categorías aumente mucho. Para evitar esta multiplicación de categorías, se introducen estructuras de rasgos para distinguir entre los miembros de las categorías verbales. Esto se logra asociando con cada verbo del lexicón un rasgo atómico, llamado *SUBCAT*, con un valor adecuado. El rasgo *SUBCAT* indica al resto de la gramática el número de argumentos que puede tener un verbo (sujeto, objeto directo, objeto indirecto). Por ejemplo: *El cartero ha traído una carta para Ana / Susan gave me a present.*

Aunque en principio la subcategorización estaba pensada exclusivamente para los verbos, muchos adjetivos y nombres tienen también marcos de subcategorización. Por ejemplo, el adjetivo *apparent* puede ir seguido por un complemento circunstancial de finalidad o por un sintagma preposicional.

Además, los verbos presentan restricciones de subcategorización no sólo en sus complementos, sino también en los sujetos, para lo que se utiliza el rasgo *SUBJECT*.

10.3.3 Dependencias de larga distancia

A veces el constituyente subcategorizado por el verbo mantiene una relación con su predicado que está a cierta distancia (*What cities does Continental service?*). Para solucionar este problema las gramáticas de unificación mantienen una *lista de huecos* o *gap list*, que se implementa como un rasgo denominado *GAP*. Este rasgo se pasa de frase a frase en el árbol de análisis. El relleno del hueco se coloca en la lista, que al final se debe unificar con el marco de subcategorización del verbo.

10.4 Implementación de la unificación

El operador de unificación coge dos o más estructuras de rasgos como entrada y devuelve una sola estructura fusionada si todo es correcto, o una señal de fallo si las entradas no son compatibles. Las estructuras de entrada se representan por un grafo dirigido acíclico, donde los rasgos son representados por etiquetas de los arcos dirigidos, y los valores son símbolos atómicos o grafos dirigidos acíclicos. Se representa, por tanto, con un algoritmo recursivo adaptado para dar cabida a los diversos requerimientos de la unificación.

10.4.1 Estructuras de datos para la unificación

Se puede considerar que la unificación tiene un aspecto destructivo de los rasgos de las estructuras que se fusionan. Para facilitar esta fusión se añaden unas aristas o campos adicionales que se utilizan para representar las estructuras de rasgos de entrada. Así, cada estructura de rasgos consiste en dos campos: el contenido y el puntero. El contenido puede estar vacío o contener una estructura de rasgos normal. El puntero puede ser nulo o contener un puntero a otra estructura de rasgos. Si el puntero del grafo es nulo, el contenido del grafo incluye la estructura que se va a procesar. Si no es nulo, el destino del puntero representa la estructura que se debe procesar. La fusión realizada en la operación de unificación se logra alterando el puntero del grafo durante el procesamiento.

10.4.2 Algoritmo de unificación

El primer paso de este algoritmo es obtener los contenidos reales de los argumentos. El siguiente es comprobar los casos de recursión antes de proceder a una llamada recursiva. Hay tres posibles casos:

- Los argumentos son idénticos.
- Uno o ambos argumentos tienen un valor nulo.
- Los argumentos son no nulos y no idénticos.

Si las estructuras son idénticas, el puntero del primero se fija con el valor del segundo y se devuelve el segundo. Esto se hace para que las posteriores unificaciones que añadan información en una estructura se integren en ambas.

Si uno de los argumentos es nulo, el puntero del argumento nulo se cambia para que apunte al otro argumento, que es el que se devuelve. El resultado es que ambas estructuras ahora apuntan al mismo valor.

Si los dos argumentos tienen valores atómicos distintos, son incompatibles y el algoritmo devuelve una señal de fallo. Si son estructuras complejas distintas, se necesita una llamada recursiva para asegurar que las partes que componen las estructuras son compatibles. Para ello se comprueba si alguna de las unificaciones es correcta, en cuyo caso se devuelve como valor de la unificación.

10.5 Análisis con restricciones de unificación

Se asocian las restricciones de unificación con las reglas de la gramática de contexto libre, y las estructuras de rasgos con los estados de búsqueda, integrándolas en un analizador. Se puede utilizar cualquier algoritmo de búsqueda.

Si se integran en el algoritmo de Earley, las estructuras de rasgos proporcionan una representación más detallada de los constituyentes del análisis, e impiden la entrada de constituyentes mal formados que violen las restricciones de unificación. Para poder utilizar este algoritmo se deben realizar cambios en las reglas y en los estados. En las reglas se añade una estructura de rasgos derivada de sus restricciones de unificación. En los estados se añade un campo que contiene el grafo que representa a la estructura correspondiente al estado, pero esto hace que el nuevo estado sea más complejo, puesto que las estructuras asociadas con los estados son también más complejas. A veces también se producen estados idénticos que pueden hacer que se duplique el trabajo.

Estos problemas se resuelven de la siguiente forma: si un nuevo estado es idéntico a otro existente (en relación a su regla, posición inicial y final, subpartes y posición •), el nuevo estado no se introduce en la tabla si su grafo ya está incluido en el grafo del estado existente.

Debido a la naturaleza destructiva del algoritmo de unificación, para poder utilizar los estados en posteriores derivaciones, se deben copiar los grafos asociados con los estados antes de unificarlos.

Otro enfoque distinto al del algoritmo de Earley consiste en utilizar un análisis basado en la unificación, de forma que en lugar de buscar las categorías de los constituyentes en los componentes de contexto libre de la regla, el algoritmo necesita mirar el rasgo CAT del grafo asociado con una regla.

11. Semántica y análisis semántico

11.1 Representación del significado

La representación del significado consta de estructuras compuestas por un conjunto de símbolos o vocabulario de representación. Cuando están organizadas de forma adecuada, estas estructuras de símbolos se utilizan para hacerlas corresponder con objetos, propiedades de objetos y relaciones entre objetos para representar una determinada situación.

11.1.1 Representaciones semánticas computacionales

La semántica de un elemento puede capturarse mediante estructuras formales que cumplan las siguientes características:

Verificabilidad: se refiere a la capacidad del sistema para determinar la verdad o falsedad de la representación semántica de acuerdo con los hechos contenidos en la base de conocimiento.

Representaciones no ambiguas: puesto que se actúa sobre el contenido semántico de las entradas lingüísticas, la representación final del significado de una entrada debe estar libre de ambigüedad.

Imprecisión o vaguedad: también supone una dificultad para representar el significado particular de una entrada, pero no da lugar a múltiples representaciones. (*Quiero comer comida italiana*. Pero exactamente, ¿qué quiere comer?).

Forma canónica: la representación debe ser la misma para entradas con formas diferentes pero igual significado. Esto complica el análisis semántico. Hay palabras que tienen varios sentidos, y algunos son sinónimos con otras palabras. El proceso de elegir el sentido correcto en un contexto se llama *desambiguación del sentido de la palabra*.

Inferencia y variables: la inferencia se refiere a la capacidad de un sistema para extraer conclusiones válidas en base a la representación del significado de las entradas y de los conocimientos que almacena. El sistema debe poder extraer conclusiones sobre la verdad de proposiciones que no están explícitamente representadas en la base de conocimientos, sino que son lógicamente derivables a partir de ella.

Expresividad: un esquema de representación del significado debe ser suficientemente expresivo para representar cualquier afirmación de interés para el dominio de la base de conocimiento.

11.1.2 Semántica del modelo teórico

Un modelo es una construcción formal que representa el estado particular de hechos en el mundo que estamos tratando de representar. Las expresiones pueden ser asignadas a los elementos del modelo. Se establece un puente entre la representación del significado y el mundo considerado.

El vocabulario de una representación del significado consta de dos partes: el vocabulario no lógico y el lógico. El *vocabulario no lógico* consta de un conjunto abierto de nombres de objetos, propiedades y relaciones que forman parte del mundo

que estamos representando (también se llaman predicados, nodos, etiquetas o enlaces). El *vocabulario lógico* consta de un conjunto cerrado de símbolos, operadores, cuantificadores, enlaces, etc., que proporcionan el significado formal para componer expresiones en un determinado lenguaje de representación del significado. Cada elemento del vocabulario no lógico se corresponde con una parte fija y bien definida del modelo. El *dominio* de un modelo es el conjunto de objetos que son parte de la situación que se representa. No todos los elementos del dominio tienen que tener un concepto correspondiente en la representación del significado, ni es necesario que sea único.

Los objetos indican elementos del dominio, las propiedades indican conjuntos de elementos del dominio y las relaciones indican conjuntos de tuplas de elementos del dominio. La función que asigna los elementos del vocabulario no lógico a su denotación en el modelo se llama *interpretación*.

11.1.3 Lógica de primer orden

Es un enfoque flexible, bien entendido y tratable computacionalmente, para representar el conocimiento que satisface muchas de las condiciones para un lenguaje de representación del significado (verificabilidad, inferencia y expresividad).

Los elementos básicos de la lógica de primer orden son:

- Término: pueden ser constantes, funciones o variables. Son una forma de llamar, o apuntar a, un objeto en el mundo considerado.
- Constantes: se refiere a objetos específicos en el mundo que se describe. Se representan por letras mayúsculas (A, B, etc.). Una constante se refiere a un solo objeto, pero los objetos pueden tener múltiples constantes que se refieren a ellos.
- Funciones: son términos que se utilizan para referirse a objetos específicos que no tienen asociada una constante.
- Variable: se representan con letras minúsculas (a, b, etc.), y permiten hacer afirmaciones y extraer inferencias sobre los objetos sin tener que hacer referencia a ningún objeto particular.
- Los predicados son símbolos que se refieren o nombran las relaciones entre un número determinado de objetos en un dominio (1, 2, etc.).
- Conectores lógicos: permiten crear representaciones más grandes enlazando fórmulas lógicas usando operadores (y, o, no).
- Cuantificadores: las variables se refieren a objetos anónimos particulares y a todos los objetos de la colección. Estos dos usos se pueden hacer gracias a los

cuantificadores. Los básicos son el existencial (\exists) y el universal (\forall). Se utilizan para sustituir un objeto por una variable. En la implicación (\Rightarrow), hay un antecedente y un consecuente. Para satisfacer una variable con cuantificador existencial, al menos una sustitución debe dar como resultado una frase cierta. Las frases con variables que tienen cuantificador universal deben ser ciertas en todas las posibles sustituciones.

- Expresiones Lambda: se utiliza para permitir expresiones de la forma $\lambda x.P(x)$, y sirve para producir nuevas expresiones lógicas donde las variables están ligadas a términos especificados. Lo que hace interesantes a las expresiones lambda son las nociones de equivalencia y reducción que se pueden definir sobre ellas. Así, la reducción λ consiste en reemplazar las variables λ con dichos términos ($\lambda x.P(x)(A)$ se convierte en $P(A)$). Aplicando la técnica *currying* se puede convertir un predicado con múltiples argumentos en una secuencia de predicados con un único argumento cada uno.

La inferencia o deducción es la capacidad de añadir nuevas proposiciones válidas a una base de conocimiento o determinar que una proposición es verdad sin que esté explícitamente contenida en la base de conocimiento. El *modus ponens* es el método de inferencia más utilizado, y se corresponde con el razonamiento siguiente:

$$\frac{\alpha}{\frac{\alpha \Rightarrow \beta}{\beta}}$$

De las fórmulas lógicas expresadas encima de la línea, se obtiene la de abajo por inferencia o deducción. El encadenamiento puede ser hacia delante o hacia atrás. En el primer, caso se añade un nuevo hecho o proposición a la base de conocimientos partiendo de los que ya están en la base. El encadenamiento hacia atrás se utiliza para demostrar que un hecho es cierto, buscando las proposiciones de la base de conocimientos que nos permitan llegar a esa conclusión.

El encadenamiento hacia delante se utiliza cuando hay pocos datos y/o muchas posibles soluciones, pero es poco específico ya que, aunque un hecho no sea necesario, será inferido y almacenado.

Por otra parte, el encadenamiento hacia atrás es útil cuando hay mucha información disponible, aunque no toda sea relevante, y a veces puede ser más eficaz. Si el consecuente de una regla es cierto, se supone que el antecedente también lo es, lo que permite razonar hacia atrás (abducción).

Ninguno de estos dos tipos de inferencia es completo, por lo que se puede utilizar otra técnica de inferencia llamada *resolución*, que es sólida y completa. Su inconveniente es que computacionalmente requiere mucho más esfuerzo que las dos anteriores. Utiliza la refutación para comprobar una sentencia, es decir, trata de llegar a una contradicción negando el hecho original, demostrando de esta forma que el hecho original es verdadero.

11.1.4 Representación de eventos y estados

Los estados son las propiedades que no cambian en el tiempo, y los eventos implican una modificación de alguna situación. Existen verbos que pueden tener un número variable de argumentos, lo que provoca que en la representación de ambos elementos surjan estos problemas:

- Determinar el correcto número de funciones de un evento.
- Representar hechos sobre funciones asociadas con un evento.
- Garantizar que todas las inferencias correctas se pueden derivar directamente a partir de la representación de un evento.
- Asegurar que no se puede derivar ninguna inferencia incorrecta a partir de la representación de un evento.

Para resolver estos problemas se utilizan los *meaning postulates*, que se basan en la idea de que los elementos léxicos se pueden definir según sus relaciones con otros elementos léxicos. Los *meaning postulates* son fórmulas que expresan algunos aspectos del sentido del predicado. Sin embargo, este enfoque presenta problemas de escalabilidad, por lo que se suele suponer que todas las fórmulas representan el mismo predicado pero en el que algunos argumentos no aparecen, siendo sustituidos por variables. Se parte de la frase con más argumentos:

Jorge subió las maletas al tren de Sevilla.	$\text{Subir}(\text{Jorge}, \text{Maletas}, \text{Tren}, \text{Sevilla})$
Jorge subió al tren.	$\exists w, x, y \text{ Subir}(\text{Jorge}, w, \text{Tren}, y)$
Jorge subió las maletas.	$\exists w, x \text{ Subir}(\text{Jorge}, \text{Maletas}, w, x)$
Jorge subió las maletas al tren.	$\exists w \text{ Subir}(\text{Jorge}, \text{Maletas}, \text{Tren}, w)$
Jorge subió al tren de Sevilla.	$\exists w \text{ Subir}(\text{Jorge}, w, \text{Tren}, \text{Sevilla})$

Pero este enfoque no permite individualizar los eventos. En el ejemplo se supone que el significado del verbo implica montarse en un medio de locomoción, pero podría tener otros: escalar una montaña, elevar el precio de un producto, ascender de categoría profesional, etc. Para solucionar este problema se pueden convertir los eventos en

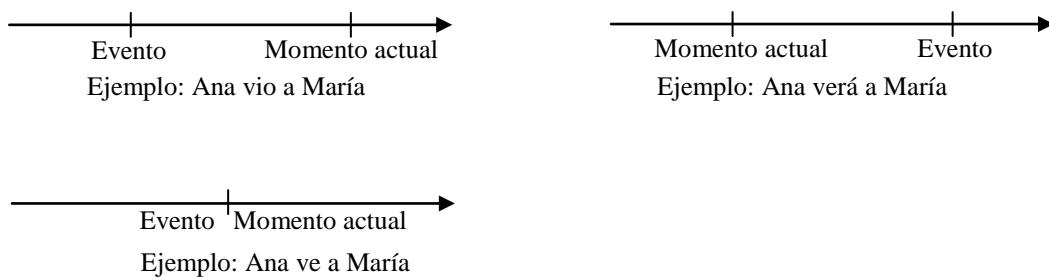
entidades cuantificables, añadiendo una variable como primer argumento en la representación, pudiendo incluso añadir nuevas afirmaciones (representación de eventos *Davidsonianos*): $\exists e \text{ Subir}(e, \text{Jorge}, \text{Maletas}, \text{Tren}, \text{Sevilla}) \wedge \text{Fecha}(e, \text{Viernes})$. Pero esta dualidad se puede eliminar utilizando relaciones adicionales (representación de eventos *neo-Davidsonianos*):

$$\exists e \text{ Subir}(e) \wedge \text{QuienSube}(e, \text{Jorge}) \wedge \text{QueSube}(e, \text{Maletas}) \wedge \text{DondeSube}(e, \text{Tren}) \wedge \text{Destino}(e, \text{Sevilla}) \wedge \text{Fecha}(e, \text{Viernes})$$

De esta forma:

- No es necesario especificar un número determinado de argumentos para el predicado, se pueden añadir cuando aparezcan en la entrada.
- Sólo se especifican las funciones mencionadas en la entrada.
- Las conexiones lógicas entre ejemplos relacionados se satisfacen sin necesidad de indicar sus significados.

En estas representaciones es importante el momento en el que se produce el evento, o más específicamente, se puede indicar si un evento *precede* a otro en el tiempo. Se pueden añadir variables temporales que representan el intervalo del evento, el final del evento, si precede al momento actual, si ocurrirá en el futuro o si el evento ocurre en este momento. Pero en inglés, a veces el presente tiene sentido de futuro, e incluso el futuro puede referirse a un evento pasado. Para solucionar este problema Reichenbach introduce la idea de *punto de referencia* [14]. El momento actual se usa como punto de referencia del evento, que puede ocurrir antes, en ese momento o después:



Otro punto importante es el aspecto, que indica si un evento ha terminado o está en curso, si ocurre en un instante determinado o en un intervalo de tiempo, y si algún estado en particular se produce a causa de él. Según esto, las expresiones de eventos pueden ser [18]:

- De estado: indica una propiedad o estado en un momento determinado. No permiten modificaciones temporales, ya que es estático (el niño está herido).
- De actividad: evento dinámico que ocurre y progresó en el tiempo. Describe un evento que no tiene un final establecido (la pelota rodó por la cuesta).
- De realización: evento dinámico delimitado en el tiempo y que progresó hacia un límite interno. Señala un evento con un punto final que da lugar a un estado concreto (la ropa se secó).
- De logro: evento dinámico delimitado, de duración muy breve y culmina en un punto concreto. El evento sucede en un instante dado y no se equipara con ninguna actividad particular que conduzca al estado (el jarrón se ha roto).

11.1.5 Lógicas descriptivas

Representan conocimiento terminológico en un dominio de forma estructurada. Describen tanto los conceptos de un dominio como la semántica que establece una equivalencia entre las fórmulas de lógicas de descripción y expresiones lógicas de primer orden. Las lógicas descriptivas se diseñaron como una extensión de los *frames* (marcos) y las redes semánticas, dotadas con una semántica formal basada en lógica. Esta semántica proporciona la posibilidad de representar la ontología de un dominio de aplicación y permitir razonar sobre él, formalizando las propiedades tanto de los individuos que pertenecen al dominio como de las relaciones entre conceptos y roles.

En el modelado de lógicas descriptivas, la base de conocimiento está formada por dos componentes: *TBox* (caja terminológica) y *ABox* (caja de afirmaciones). La primera contiene sentencias que describen conceptos jerárquicos (relaciones entre conceptos), mientras que en la *ABox* se incluyen afirmaciones sobre los individuos en la jerarquía (relaciones entre personas y conceptos). Esta distinción entre *TBox* y *ABox* permite describir y formular procedimientos de decisión e inferencias.

Por lo tanto, las lógicas descriptivas son sistemas formales que representan conocimiento, es decir, no son simplemente un almacén de terminología y descripciones, sino que infieren hechos implícitos a partir de los datos. Un desafío importante en este tipo de lógicas es encontrar algoritmos de decisión para los problemas de inferencia (satisfacibilidad, subsunción y consistencia), que tengan la menor complejidad posible y sean abordables computacionalmente.

La subsunción, como una forma de inferencia, implica determinar si existe un conjunto de relaciones entre dos conceptos, sobre la base de los hechos afirmados en *TBox*. La

comprobación de instancias, por su parte, pregunta si un individuo es miembro de una categoría según los hechos conocidos que se encuentran en *ABox*.

El uso más destacado de las lógicas descriptivas actuales es el *Lenguaje de Ontologías Web* (OWL), que es un lenguaje de marcado utilizado para publicar y compartir datos empleando ontologías en la Web. Está construido sobre RDF (modelo de datos para metadatos) y codificado en XML, y se utiliza en la especificación de la Web semántica.

11.2 Semántica computacional

11.2.1 Análisis semántico dirigido por sintaxis

Está basado en el principio de composicionalidad, según el cual el significado de las expresiones complejas está completamente determinado por su estructura (ordenación, agrupación y relaciones entre componentes) y por el significado de sus componentes. Por lo tanto, en el análisis semántico dirigido por sintaxis, la composición de las representaciones del significado es guiado por los componentes y relaciones sintácticas proporcionadas por las gramáticas formales estudiadas en el apartado 2 de este trabajo.

Para poder representar el significado correspondiente a una frase se requiere gran cantidad de conocimiento específico sobre el contenido de dicha frase, además de su árbol de análisis sintáctico. Esto representa un grave problema, ya que existen un número infinito de árboles de análisis que representan a una gramática. Por ese motivo los lenguajes no se definen como la enumeración de dichos árboles, sino como la especificación de un número finito de dispositivos capaces de generar el conjunto de salidas deseado: las reglas de la gramática y las entradas léxicas. Esto se conoce como *hipótesis regla-a-regla*.

11.2.2 Ampliación semántica para las reglas sintácticas

Se utilizan tres técnicas para instanciar el enfoque *regla-a-regla* para el análisis semántico:

- Asociar funciones del tipo expresión- λ con elementos léxicos de forma compleja.
- Copiar valores semánticos de los hijos a los padres en reglas sin ramificación.
- Aplicar la semántica de uno de los hijos de una regla a la semántica de otro hijo de la regla utilizando reducción λ .

Estas tres técnicas sirven para demostrar una división general del trabajo que se realiza para diseñar adjuntos semánticos en este marco composicional. Las reglas léxicas introducen cuantificadores, predicados y términos en nuestras representaciones del

significado. Los adjuntos semánticos para las reglas de gramática ponen estos elementos juntos de la forma correcta, pero generalmente no introducen nuevos elementos en las representaciones que se crean.

11.2.3 Ambigüedad y subespecificación en el ámbito de los cuantificadores

Tanto la ambigüedad como la subespecificación son dos fenómenos que reflejan descripciones incompletas. Para resolver la ambigüedad en el ámbito de los cuantificadores (saber cuál tiene el ámbito más externo), se necesitan las siguientes capacidades:

- Posibilidad de crear eficazmente representaciones subespecificadas que expresen todas las posibles lecturas sin enumerarlas explícitamente.
- Una forma de generar, o extraer, todas las posibles lecturas a partir de esta representación.
- Capacidad de elegir entre las posibles lecturas (requiere el uso del contexto y conocimiento del mundo, por lo que no es fácil de conseguir).

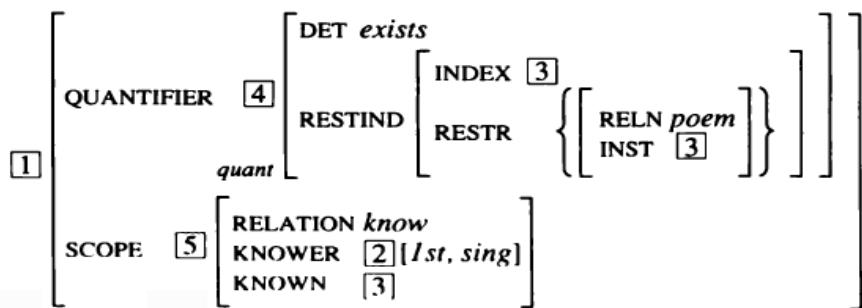
Enfoques de almacenar y recuperar: una representación subespecificada del significado de una frase especificaría lo que sabemos sobre cómo se combinan sus representaciones, pero no sobre la colocación de los cuantificadores en la representación final. Para proporcionar esta última funcionalidad se utiliza la idea del *almacenamiento de Cooper* [4] y las expresiones λ . El almacén incluye una representación del significado nuclear junto con una lista indexada de expresiones cuantificadas (expresiones λ), obtenida de los nodos que están por debajo en el árbol. Para recuperar una representación totalmente especificada del almacén, se elige uno de los elementos y se aplica a la representación utilizando una reducción λ .

Enfoques basados en las restricciones: el enfoque anterior presenta dos problemas. El primero es que sólo resuelve el problema de las ambigüedades del ámbito introducidas por sintagmas nominales cuantificados, no por otras construcciones sintácticas ni otros elementos léxicos. El segundo es que, aunque permite enumerar todos los ámbitos posibles de una expresión, no permite imponer restricciones adicionales sobre estos ámbitos. Estas restricciones son esenciales para aplicar conocimiento específico en los aspectos léxico, sintáctico y pragmático, reduciendo el rango de posibilidades de una expresión. En lugar de centrarse en cómo recuperar las representaciones del almacén, debemos centrarnos en cómo interpretar eficazmente las representaciones

subespecificadas, incluyendo restricciones que la representación final debe satisfacer. El enfoque *hole semantics* es uno de los que utilizan esta perspectiva basada en restricciones. Utiliza tuplas que constan de un conjunto de entidades (*agujeros* y etiquetas), un conjunto de representaciones etiquetadas y un conjunto de restricciones de ámbito. Los agujeros representan ámbitos no asignados y pueden ser tapados por etiquetas, siempre que no se infrinja ninguna restricción. Las restricciones de ámbito se interpretan como relaciones de dominancia en nodos de representaciones semánticas etiquetadas (es decir, fórmulas de primer orden vistas como árboles). Posteriormente se llenan todos los agujeros con expresiones etiquetadas de tal forma que representan todas las restricciones. Las representaciones subespecificadas se pueden construir utilizando enfoques basados en la unificación o basados en lambda, y también con lógica lineal.

11.2.4 Enfoques basados en unificación para el análisis semántico

Para llevar a cabo el proceso de construcción de las representaciones semánticas se necesita una sintaxis completamente especificada. Supongamos que el análisis sintáctico está basado en una colección de categorías sintácticas, cuya interrelación se describe en las reglas de estructura de la frase, y cuyo contenido se representa en las entradas léxicas. Las categorías se representan como estructuras de rasgos, en las que la característica semántica (*sem*) consta de un *index* que designa la variable que se debe equiparar con otra parte de la información. Los determinantes tienen, además, las características *restr* (restricción) y *scope* (ámbito nuclear), que se encargan de asegurar la colocación correcta de estos dos componentes principales de la representación:



El enfoque basado en la unificación se puede aplicar para tratar un amplio rango de fenómenos semánticos y es muy eficiente. Pero no hace distinción entre las variables usadas para la unificación y las variables usadas en las representaciones semánticas. Esto provoca problemas cuando se tiene que copiar una parte de la representación semántica, por ejemplo en la coordinación de frases.

11.2.5 Integración de la semántica en el analizador de Earley

Los pasos son los mismos del algoritmo de unificación, pero con las siguientes modificaciones:

- Las reglas tienen un nuevo campo para contener sus adjuntos para la semántica.
- Los estados del grafo tienen un nuevo campo para mantener la representación del significado del constituyente.
- La función para poner en cola se altera de modo que, cuando se introduce un estado completo en el grafo, su semántica es calculada y almacenada en el campo semántico del estado.

La unificación de la estructura de rasgos se tiene que realizar antes que el análisis semántico para que éste se lleve a cabo sólo sobre árboles válidos y para que las características necesarias estén presentes. Una ventaja de este enfoque integrado sobre el enfoque secuencial radica en que, si se encuentra una mala formación semántica en la representación del significado que está siendo creado, el estado correspondiente se puede bloquear desde la entrada del grafo.

Uno de los inconvenientes de la integración de la semántica directamente en el analizador es que hay que realizar un gran esfuerzo en el análisis semántico para tratar los constituyentes que no tienen padre, ya que al final no contribuyen a un análisis exitoso.

11.2.6 Modismos y composicionalidad

La composicionalidad presenta problemas cuando se examina el lenguaje real. En muchos casos el significado de un constituyente no es la suma de los significados de sus partes. La forma más sencilla de gestionar las construcciones idiomáticas es introducir nuevas reglas gramaticales específicas que mezclan elementos léxicos con constituyentes gramaticales e introducen contenido semántico que no se deriva de ninguna de sus partes. En este caso, un analizador que siga el algoritmo de Earley producirá dos árboles de análisis, uno que representa el modismo y otro que representa el significado composicional. Resumiendo, gestionar modismos requiere al menos los siguientes cambios en el marco composicional general:

- Permitir la mezcla de elementos léxicos con constituyentes gramaticales tradicionales.
- Permitir la creación de constituyentes adicionales específicos de los modismos para controlar la producción de modismos.

- Permitir adjuntos semánticos que introducen términos y predicados lógicos no relacionados con ningún constituyente de la regla.

11.3 Semántica léxica

11.3.1 *Sentido de las palabras*

El significado de la forma canónica de una palabra (lema) puede variar según el contexto. El sentido de una palabra es una representación de un aspecto de su significado. Los distintos sentidos de una palabra pueden no tener relación entre ellos, en cuyo caso se dice que los sentidos son homónimos, y la relación se llama homonimia. En cambio, cuando los sentidos están relacionados semánticamente, la relación que existe entre los significados se llama polisemia. Existe un subtipo de polisemia, llamado metonimia, que consiste en el uso de un aspecto de un concepto o entidad para referirse a otros aspectos de la entidad o a la entidad en sí misma (un Picasso, un Rioja, se tomó una copa, etc.). A veces es difícil saber si dos sentidos son homónimos o polisémicos. Una forma práctica de determinar si dos sentidos son distintos es utilizarlos en una sola frase. Esta conjunción de lecturas antagónicas se llama zeugma.

Los homónimos tienen la misma pronunciación y ortografía, pero los homófonos son sentidos que están enlazados a lemas que se pronuncian igual pero su ortografía es diferente (ant/aunt, plain/plane, acerbo/acervo, abría/habría, asta/hasta). Los homófonos provocan problemas en el reconocimiento de la voz.

11.3.2 *Relaciones entre sentidos*

- Sinonimia y antonimia: cuando dos palabras tienen sentidos idénticos o casi idénticos se dice que son sinónimos o que tienen el mismo significado proposicional. Cuando los sentidos son opuestos, se dice que son antónimos.
- Hiponimia: un sentido es hipónimo de otro si el primero es más específico, es una subclase del segundo (mesa es hipónimo de mueble). Por el contrario, mueble es hiperónimo de mesa. Si el sentido de A es un hipónimo del sentido de B, se puede establecer la relación $\forall x A(x) \Rightarrow B(x)$. La hiponimia suele ser una relación transitiva. Una *taxonomía* es una disposición particular de los elementos de una ontología en una estructura de árbol, que se refiere a una relación de hiperonimia entre dos sentidos.

- Campos semánticos: meronimia es la relación parte-todo. Una *rueda* es un meronimo de *coche*, y un *coche* es holonimo de *rueda* (relación binaria). Un campo semántico es un modelo de una relación integrada entre conjuntos de palabras de un dominio (relación múltiple).

11.3.3 WordNet: una base de datos de relaciones léxicas

WordNet consta de tres bases de datos separadas: una para nombres, otra para verbos y la tercera para adjetivos y adverbios. Cada base de datos consta de un conjunto de *lemmas*, anotados con un conjunto de sentidos. Los sinónimos de cada sentido guardado en WordNet se llaman *synset*. Cada concepto se representa con listas de sentidos de palabras que se pueden usar para expresar el concepto. Cada *synset* está relacionado con su *synset* más general y con el más específico a través de relaciones de hiperonimia e hiponimia.

11.3.4 Participantes de los eventos

Los roles temáticos tratan de capturar la concordancia semántica entre los participantes en un evento. Los sujetos de los verbos se llaman **agentes**, y el rol temático del agente representa cierto grado de voluntad. El rol temático del objeto directo del verbo es el **tema**. Los roles temáticos y semánticos permiten realizar inferencias que no se pueden hacer con una simple cadena de palabras o incluso con árboles de análisis sintáctico.

La diátesis es una categoría gramatical que describe el número de argumentos verbales obligatorios para un predicado verbal. Las alternancias de diátesis o alternancias verbales son las distintas estructuras de argumentos que se pueden dar en un verbo según su rol:

Juan (*agente*) dio el libro (*tema*) a Ana (*paciente*)

Juan (*agente*) dio a Ana (*paciente*) el libro (*tema*)

Pero existen problemas con los roles temáticos, relacionados con la dificultad de establecer un conjunto estándar de funciones y de elaborar una definición formal de los papeles de agente, tema o instrumento (pueden dividirse en roles más específicos). También se presenta la dificultad de definir los roles semánticos (los casos de agentes pueden ser: animado, volitivo, de sentimiento o causal, pero no todos los sintagmas nominales tienen estas propiedades).

Una solución a estos problemas es el modelo de **roles semánticos generalizados** (proto-agente, proto-paciente), que definen roles con un conjunto de funciones heurísticas que suelen acompañar a sus significados. Así, cuantas más propiedades

asociadas al rol tenga un argumento, más probabilidad tendrá de poder ser etiquetado con dicho rol.

Además de usar proto-roles, algunos modelos solucionan los problemas referidos a los roles temáticos definiendo roles semánticos específicos para un verbo o para un conjunto de sintagmas verbales. Por ejemplo, **ProBank** utiliza proto-roles y roles semánticos específicos para los verbos, y **FrameNet** utiliza roles semánticos específicos de un marco. **ProBank** es útil para la recuperación de información semántica sobre argumentos verbales referidos a un verbo específico. **FrameNet** utiliza marcos (estructuras similares a scripts), que crean instancias para un conjunto de roles semánticos específicos del marco, llamados **elementos del marco** (roles semánticos del marco). También establecen relaciones entre marcos y elementos. Además pueden generalizar, establecer relaciones de causalidad, etc. Los roles semánticos proporcionan una forma de expresar la semántica de un argumento en relación con el predicado.

Una **restricción selectiva** es un tipo de restricción semántica que un verbo impone a los conceptos que pueden ser definidos como sus argumentos. Se asocia a un sentido, no a un lexema, y puede ayudar en la desambiguación del sentido de una palabra. Para representar las restricciones de selección se puede utilizar la representación neo-Davidsoniana:

$$\exists e, x, y \text{ Comer}(e) \wedge \text{Agente}(e, x) \wedge \text{Tema}(e, y) \wedge \text{CosaComestible}(y)$$

Pero el uso directo de estas restricciones presenta dos problemas. Primero, usar lógica de primer orden para realizar la tarea simple de hacer cumplir las restricciones selectivas es excesivo, por el coste computacional que supone. Segundo, este enfoque implica disponer de una gran base de conocimientos de hechos sobre los conceptos para hacer restricciones de selección.

Una forma más práctica de hacer este tipo de restricciones es utilizar *synsets* en lugar de conceptos lógicos. Cada predicado especifica un *synset* como restricción de selección en cada uno de sus argumentos. Un rol no necesita ser exactamente igual a la restricción del *synset*, sólo debe tener a dicho *synset* como uno de sus superiores.

11.4 Semántica léxica computacional

Las palabras con significados similares suelen aparecer en contextos similares, ya sea en corporas o en diccionarios y tesauros. Por lo tanto, la similitud del contexto es importante para detectar la similitud semántica. Esta última tiene un papel importante en recuperación de información, búsqueda de respuestas, generación de resúmenes, clasificación de textos, clasificación automática de documentos y detección de plagios.

11.4.1 Desambiguación del sentido de las palabras

Los analizadores semánticos suelen ignorar la ambigüedad léxica. Pero debemos contar con algún medio de selección del sentido correcto de las palabras de entrada, porque de lo contrario la cantidad de homonimia y polisemia que contiene el léxico impediría alcanzar una interpretación correcta.

La desambiguación del sentido de las palabras (WSD, Word Sense Disambiguation) tiene la capacidad de mejorar muchas tareas de procesamiento del lenguaje natural. Los algoritmos de WSD cogen como entrada una palabra en su contexto junto con un número fijo de posibles sentidos, y devuelven como salida el sentido correcto de la palabra para ese contexto.

Se pueden distinguir dos variantes de la tarea genérica de WSD. En la tarea **muestra léxica** (lexical sample), se intenta desambiguar el sentido de unas pocas palabras escogidas previamente. Debido al reducido tamaño de la muestra, se pueden utilizar enfoques de aprendizaje automático, entrenando a los sistemas clasificadores con estas muestras.

Por el contrario, en la tarea **todas-las-palabras** (all-words), se trata de desambiguar el sentido de todo el texto. Por el gran número de etiquetas que se utilizan se presenta el problema de escasez de datos, ya que es imposible que haya datos de entrenamiento para todas las palabras del corpus de pruebas. Además, si el tamaño del léxico es suficientemente grande, la polisemia que aparece provoca que los enfoques basados en este tipo de entrenamiento, utilizando todos los términos, sea impracticable.

12. Conclusiones

Las gramáticas de contexto libre (CFG), también llamadas generativas, explican el procesamiento de formaciones lingüísticas de manera simple y eficiente. Consisten en un conjunto de reglas o producciones expresadas mediante símbolos terminales y no terminales. Puesto que se basan en un formalismo declarativo, las CFG no especifican cómo calcular el árbol de análisis sintáctico de una frase. Por tanto, un analizador

sintáctico explora todos los posibles árboles para encontrar el árbol sintáctico correcto. Esta búsqueda se puede realizar construyendo el árbol desde el nodo raíz hacia las hojas (top-down), o comenzando en las hojas hacia arriba hasta el nodo raíz (bottom-up).

La ambigüedad generalizada en las gramáticas junto con la construcción repetida de árboles sintácticos lleva a una gran ineficiencia de los enfoques de retroceso en el análisis sintáctico. La ambigüedad estructural aparece cuando se asigna más de un árbol sintáctico a una frase, y puede ser adjunta o de coordinación. En este trabajo se detallan tres algoritmos de programación dinámica que se utilizan para tratar este problema: CKY, Earley y análisis gráfico.

Las CFG tienen ciertas limitaciones: multiplicación de las categorías primitivas, tratamiento insatisfactorio de los componentes discontinuos e imposibilidad de modelar todos los fenómenos gramaticales presentes en los lenguajes naturales. Las teorías basadas en la unificación defienden que las lenguas naturales se pueden describir con CFGs aumentadas con extensiones formales, las estructuras de rasgos, que permiten enriquecer el poder expresivo de las CFGs. La unificación es un proceso que combina la información de dos o más estructuras de rasgos para formar una nueva que contenga toda la información de las primeras, siempre y cuando sean compatibles.

Las reglas independientes del contexto pueden permitir combinaciones agramaticales, por lo que es necesario imponer restricciones para representar información sobre la concordancia en número y persona, la subcategorización y las categorías semánticas. Además, modelan fenómenos complejos, pueden calcular eficientemente la semántica para las representaciones sintácticas y solucionan algunos de los problemas de sobre-generación de etiquetas.

En cuanto al significado que representa al modelo del conocimiento, consta de estructuras compuestas por un conjunto de símbolos o vocabulario de representación. Cuando están organizadas de forma adecuada, estas estructuras de símbolos se utilizan para hacerlas corresponder con objetos, propiedades de objetos y relaciones entre objetos para representar una determinada situación. Esta representación del conocimiento debe cumplir ciertos requisitos para poder procesarla, como ser capaz de determinar la verdad de las proposiciones, permitir representaciones no ambiguas, posibilitar la inferencia y ser suficientemente expresiva. La lógica de primer orden es un enfoque flexible y tratable computacionalmente para representar este conocimiento, mediante la captura de estados y eventos del modelo.

El análisis semántico está basado en el principio de composicionalidad, según el cual el significado de las expresiones complejas está completamente determinado por su estructura (ordenación, agrupación y relaciones entre componentes) y por el significado de sus componentes. Se utilizan expresiones λ y términos complejos para ampliar la lógica de primer orden con el objetivo de representar la composicionalidad. Sin embargo, los cuantificadores introducen un tipo de ambigüedad que es difícil de resolver mediante la composicionalidad. La subespecificación, al igual que la ambigüedad, es un fenómeno que refleja descripciones incompletas, pero se puede utilizar para resolver de forma eficaz las múltiples interpretaciones del significado que se producen por las ambigüedades y por los modismos.

Por último, la semántica léxica estudia el significado de las palabras y las relaciones entre ellas, así como la variación semántica según el contexto. El sentido de una palabra es una representación de un aspecto de su significado. Entre los distintos sentidos de una palabra se pueden establecer varios tipos de relaciones: hominimia, polisemia, sinonimia, etc. Los campos semánticos capturan estas conexiones semánticas entre los lexemas que representan el significado en un dominio. Los roles temáticos describen la relación semántica que los argumentos tienen con respecto al predicado, es decir, la concordancia semántica entre los participantes de un evento. Un verbo puede imponer una restricción selectiva a los conceptos que pueden actuar como sus argumentos, ayudando a desambiguar el sentido de una palabra.

Glosario de términos

Verificabilidad: se refiere a la capacidad del sistema para determinar la verdad o falsedad de la representación semántica de acuerdo con los hechos contenidos en la base de conocimiento. Consiste en confrontar una hipótesis o argumento con los hechos y ver si válidos de forma lógica.

Forma canónica: representación ortográfica estándar de una palabra. Varias unidades léxicas pueden corresponder a una única forma canónica. Por ejemplo, subir es la forma canónica de subiendo, subiría, subirás, etc.

Expresividad: capacidad del esquema de representación del significado para manifestarse con suficiente elocuencia y exactitud en la interpretación de afirmaciones de interés para el dominio de la base de conocimiento.

Modelo: construcción formal que representa el estado particular de hechos en el mundo que se está tratando de representar.

Vocabulario lógico: conjunto cerrado de símbolos, operadores, cuantificadores, enlaces, etc., que proporcionan el significado formal para componer expresiones en un determinado lenguaje de representación del significado.

Dominio de un modelo: conjunto de objetos que son parte de la situación que se representa. No todos los elementos del dominio tienen que tener un concepto correspondiente en la representación del significado, ni es necesario que sea único.

Interpretación: función que asigna una notación apropiada a cada elemento no lógico del vocabulario en la representación del modelo. Un vocabulario no lógico consta de un conjunto abierto de nombres de objetos, propiedades y relaciones que forman parte del mundo que estamos representando.

Lógica de primer orden: enfoque flexible, bien entendido y tratable computacionalmente, para representar el conocimiento que satisface muchas de las condiciones para un lenguaje de representación del significado (verificabilidad, inferencia y expresividad). Consta de términos, constantes, funciones, variables, predicados, conectores lógicos y cuantificadores.

Sentido de una palabra: representación de un aspecto de su significado. Una palabra puede tener varios sentidos o significados.

Polisemia: pluralidad de sentidos de una palabra que están relacionados semánticamente. Por ejemplo, la palabra *canal* puede referirse a un *canal de televisión*, *canal de navegación*, *canal de agua*, etc., referidos a un conducto o vía por la que circula algo. La palabra *libro* (*book*) se puede referir a un libro editado, un libro de cuentas, etc.

Homonimia: distintos sentidos de una palabra que no tienen relación entre ellos. Por ejemplo, *bota* tiene dos significado, uno referido a calzado y otro para guardar vino. La palabra *cubo* puede referirse a un recipiente, a una operación matemática o a una figura geométrica.

Hiponimia: se produce cuando una palabra (hipónimo) posee todos los rasgos semánticos de otra más general, que es su hiperónimo, pero añade otras características semánticas que la diferencian. Por ejemplo, *tenis*, *futbol*, *baloncesto*, etc. son hipónimos de *deporte*. Las palabras *ingeniero*, *médico*, *abogado*, etc. son hipónimos de *profesión*, que es su hiperónimo.

Campo semántico: es un modelo de una relación integrada entre conjuntos de palabras de un dominio (relación múltiple). Este conjunto de palabras tiene significados relacionados, ya que comparten un rasgo semántico común, y se diferencian por otra serie de rasgos. Entre estas palabras se pueden establecer relaciones de hiponimia, meronimia, sinonimia, etc. Por ejemplo, *plato*, *cuchara*, *vaso*, *frigorífico*, *sartén*, etc. son palabras que forman parte del campo semántico *cocina*.

Rol temático: trata de capturar la concordancia semántica entre los participantes en un evento. Describe la relación semántica que los argumentos tienen con respecto al predicado. El rol temático *agente* inicia la acción del verbo y es capaz de actuar con voluntad (suele ser el sujeto), y el rol temático *paciente* es la entidad que padece el efecto de alguna acción, muchas veces pasando por un cambio de estado (suele ser el objeto directo). Los roles temáticos y semánticos permiten realizar inferencias que no se pueden hacer con una simple cadena de palabras o incluso con árboles de análisis sintáctico.

Alternancia verbal: distintas estructuras de los argumentos verbales según su rol, es decir, son las diferentes asociaciones o correlaciones que se pueden establecer entre las funciones semánticas de los argumentos exigidos por el verbo y las funciones sintácticas que éstos desempeñan.

El profesor (agente) enseñó la lección (tema) a los alumnos (paciente)

El profesor (agente) enseñó a los alumnos (paciente) la lección (tema)

El profesor (agente) enseñó la lección (tema)

El profesor (agente) enseñó a los alumnos (paciente)

La lección (tema) fue enseñada por el profesor (agente)

Restricciones selectivas: son un tipo de restricción semántica que un verbo impone a los conceptos que pueden ser definidos como sus argumentos. Se asocia a un sentido, no a un lexema, y puede ayudar en la desambiguación del sentido de una palabra.

PARTE III. DISCURSO, EXTRACCIÓN DE INFORMACIÓN Y RESÚMENES

Resumen. El análisis del discurso es una disciplina que estudia sistemáticamente la forma en que se utiliza la lengua como evento de comunicación e interacción. Este proceso de diálogo, para que sea efectivo, debe establecer relaciones de coherencia y cohesión. La desambiguación del discurso se complica cuando en el texto aparecen correferencias, palabras específicas de un dominio o se hace referencia a distintos contextos. En el presente trabajo se realiza un análisis de los métodos para detectar entidades nombradas y sus relaciones, así como el estudio de los eventos temporales. Tanto en su enfoque supervisado como en el no supervisado, estos elementos se utilizan para resolver problemas de coherencia, cohesión y correferencia en el discurso, tarea fundamental en el proceso de extracción de resúmenes y búsqueda de respuestas (Question Answering). Para valorar la calidad de la información extraída y de los resúmenes realizados a partir de ella, se deben evaluar los resultados en relación con la calidad, precisión, exhaustividad y adecuación a la necesidad de información del usuario.

13. Introducción

El lenguaje no consiste en un conjunto de frases inconexas, sino en grupos de frases colocados uno al lado de otros, de forma estructurada y coherente. A esta organización del texto se le llama discurso. En un monólogo, el hablante y el oyente se comunican en una sola dirección. En el diálogo, cada participante toma el rol de hablante y oyente por turnos. El diálogo consta de varios actos comunicativos: hacer preguntas, dar respuestas, hacer correcciones, etc. El diálogo puede ser humano-humano o humano-máquina mediante un agente conversacional.

La desambiguación del discurso es difícil de realizar. La *resolución de correferencias* trata de decidir qué pronombres hacen referencia a un sintagma nominal a lo largo del documento. Esta es una tarea muy importante en la extracción de información, la extracción de resúmenes y los agentes conversacionales.

Las *relaciones de coherencia* determinan las estructuras de conexión entre las distintas frases que forman parte del discurso. La coherencia se basa tanto en el significado expresado en las oraciones del discurso como en las relaciones de referencia entre los hechos que se tratan en el mismo. Por lo tanto, el establecimiento de la coherencia del discurso engloba un proceso de interacción entre dimensiones semánticas y pragmáticas.

Detectar relaciones de coherencia es muy útil en las tareas de clasificación automática que miden la calidad del texto. La clasificación automática asigna una calificación a los documentos mediante la medida de la coherencia interna y la comparación del contenido con documentos etiquetados como de alta calidad.

La estructura del discurso y la correferencia están relacionadas de forma que determinar la estructura del discurso puede ayudar a determinar la correferencia.

Las *relaciones de coherencia* pueden establecer distintos tipos de conexión de significado entre las expresiones yuxtapuestas: de explicación o causa, certeza, consecuencia, etc.

A continuación, en el apartado 14 se hace un breve análisis del discurso. El apartado 15 se centra en el reconocimiento de entidades nombradas como parte del proceso de extracción de información. El apartado 16 hace referencia a la búsqueda de respuestas (Question Answering) y en el 17 se detallan los sistemas de extracción de resúmenes. Por último, en el apartado 18 se presentan las conclusiones de este trabajo.

14. Discurso

Muchos tipos de texto están asociados con estructuras convencionales determinadas. Por ejemplo, los artículos académicos (resumen, introducción, metodología, resultados, conclusión), una noticia (titular, encabezamiento o resumen, episodio, etc.), los informes médicos (datos personales del paciente, antecedentes de enfermedades, síntomas, etc.).

14.1 Segmentación del discurso

Existen algunos algoritmos para dividir el discurso en partes. Uno de los más simples consiste en separar un documento en secuencias lineales de subtemas. Los algoritmos de segmentación no son capaces de encontrar la estructura jerárquica compleja, pero la segmentación lineal se utiliza en los algoritmos para recuperación de información, extracción de resúmenes y extracción de información. Las claves para realizar esta división lineal del discurso son:

- La cohesión léxica: relaciones entre palabras de las unidades léxicas, como palabras idénticas, sinónimos o hiperónimos, o la resolución de anáforas.
- Los marcadores del discurso o frases de referencia: un marcador del discurso es una palabra o frase que funciona como indicador para la estructura del discurso. Suelen ser específicos del dominio.

14.2 Referencias

Las referencias son relaciones que se establecen entre unidades o expresiones lingüísticas y entidades creadas en el discurso, de forma que designan o representan a dichas entidades.

Hay cinco tipos de expresiones de referencia:

- Sintagmas nominales indefinidos: hacen referencia a una entidad que es nueva para el oyente.
- Sintagmas nominales definidos: aluden a una entidad identificable por el oyente (ha sido mencionada antes).
- Pronombres: son un tipo de referencia definida, pero normalmente mencionan entidades presentadas una o dos frases antes en el discurso.
- Pronombres demostrativos: pueden aparecer solos o como determinantes.
- Nombres de personas, organizaciones o lugares: se pueden utilizar para aludir a entidades nuevas o ya conocidas en el discurso.

15. Extracción de información

Proceso mediante el que la información no estructurada de un texto se convierte en datos estructurados, que se pueden almacenar en bases de datos relacionales.

15.1 Reconocimiento de entidades nombradas

El primer paso en la mayoría de sistemas de extracción de información es la detección y clasificación de entidades nombradas en un texto. Una entidad nombrada es todo lo que se puede denominar con un nombre propio. El proceso de reconocer una entidad nombrada busca unidades de texto que expresan nombres propios y los clasifica según su tipo.

Los sistemas genéricos de reconocimiento de entidades nombradas se centran en la detección de nombres de personas, lugares y organizaciones. Existen aplicaciones especializadas, capaces de reconocer otros tipos de entidades según el dominio de conocimiento.

Otros tipos de entidades que se pueden reconocer para extraerlas del resto de información son las expresiones temporales (fechas, horas, eventos) y las expresiones numéricas (medidas, tantos por ciento, precios, etc.).

En la detección de entidades nombradas se pueden presentar dos tipos de ambigüedad. El primero surge por el hecho de que un mismo nombre se puede referir a distintas entidades del mismo tipo, y se resuelve mediante referencia. El segundo se plantea por el hecho de que un nombre puede hacer referencia a entidades de tipos distintos (“Rosa” puede ser un nombre o una flor).

A continuación, se debe etiquetar cada palabra del texto con las etiquetas I (Inside), B (Beginning) y O (Outside) y se selecciona un conjunto de características para asociarlas

con cada ejemplo de la entrada: items léxicos, raíz de items léxicos, forma (mayúscula, minúscula, números, signos de puntuación, etc.), afijos, etiquetado, etiquetas sintácticas de fragmentos, etc. Existen listas de entidades nombradas para personas, lugares y organizaciones que pueden servir para predecir la etiqueta de una entidad. También la presencia de palabras predictivas y N-gramas puede ayudar.

Una vez que se ha etiquetado con estas características un corpus de entrenamiento, un clasificador secuencial puede ser entrenado para etiquetar nuevas frases, mediante HMMs o MEMMs.

La evaluación de los sistemas de reconocimiento de entidades nombradas se hace con medidas de exhaustividad, precisión y F_1 :

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2P + R}$$

Valores de $\beta > 1$ se utilizan cuando el sistema requiere mayor exhaustividad, y $\beta < 1$ si la precisión es más importante. Si $\beta = 1$ están equilibradas. Es importante distinguir las medidas del rendimiento de la aplicación de las del entrenamiento.

15.2 Detección y clasificación de relaciones entre entidades

Existen relaciones genéricas como *es-parte-de*, *es-empleado*, *es-madre-de*, *es-portavoz-de*, etc. Una relación consta de un conjunto ordenado de tuplas sobre elementos de un dominio. Esta tarea forma parte también del reconocimiento de entidades.

El análisis de las relaciones se puede llevar a cabo mediante enfoques de:

- Aprendizaje supervisado: el texto es anotado con relaciones seleccionadas a partir de un pequeño conjunto realizado por analistas humanos, que sirven para entrenar al sistema. Esta tarea se divide en dos: detectar las relaciones entre entidades y clasificarlas, utilizando árboles de decisión, naive Bayes o máxima entropía. Para identificar características útiles a la hora de clasificar las relaciones se pueden obtener datos de las características de las entidades nombradas, de las palabras del texto o de la estructura sintáctica de la frase.
- Semi-supervisado: si no tenemos una gran colección de material previamente anotado, se puede optar por expandir los patrones utilizados para capturar información, aumentando la cobertura de relaciones que se pueden identificar. Se puede hacer manualmente, por analistas humanos, o automáticamente, utilizando patrones semilla y la técnica de *bootstrapping*. Surgen algunos problemas:

representar los patrones de búsqueda, valorar la exactitud y exhaustividad de los patrones descubiertos (que no provoquen desviaciones semánticas), valorar la fiabilidad de las tuplas descubiertas. Los patrones son representados de forma que capturan: el contexto antes de la primera aparición de la entidad, el contexto entre dos entidades, el contexto después de la segunda aparición de la entidad y el orden de los argumentos en el patrón.

Existen dos métodos para evaluar los sistemas de análisis de relaciones. El primero se centra en estudiar si los sistemas pueden encontrar y clasificar todas las relaciones que aparecen en un texto, para lo que se utilizan medidas de exhaustividad, precisión y F_1 , tanto de las relaciones etiquetadas como de las no etiquetadas (habilidad para detectarlas). El segundo se basa en el número de tuplas que hay cuando el sistema ha finalizado (se utiliza sobre todo en métodos no supervisados). El problema de este segundo enfoque es la exhaustividad, ya que es difícil buscar manualmente todas las relaciones que pueden aparecer, por ejemplo, en la Web.

15.3 Procesamiento temporal y de eventos

Es muy importante en aplicaciones como la búsqueda de respuestas y la extracción de resúmenes. También se utiliza para establecer un orden parcial en los eventos y expresiones temporales de un texto. Los métodos utilizados en el procesamiento de los eventos temporales son:

- Reconocimiento de expresiones temporales: se refieren a momentos de tiempo absoluto (fechas de calendario, momentos del día), relativos (a week from last Tuesday), duraciones (segundos, minutos, días, semanas, lustros, etc.) y conjuntos de estas expresiones. El proceso de reconocimiento consiste en encontrar el principio y el final de todos los fragmentos de texto que se corresponden con las expresiones temporales. Para ello se pueden utilizar sistemas basados en reglas (utilizan autómatas en cascada), clasificadores secuenciales estadísticos (etiquetas I, O y B como en las entidades nombradas), o clasificación basada en constituyentes (combina las dos anteriores, separando la clasificación de la segmentación). El problema de estas técnicas es conseguir una cobertura razonable sin generar falsos positivos (1984 cuenta la historia de Winston Smith y su degradación por el estado totalitario en el que vive).
- Normalización: consiste en asignar una expresión temporal a un punto específico en el tiempo o a una duración, siguiendo el estándar ISO 8601. Debido al cálculo de

expresiones λ , se necesitan reglas para: expresiones temporales completas, absolutas y relativas, y para duraciones.

- Detección y análisis de eventos: un evento expresa un estado que se puede asignar a un momento particular o a un intervalo de tiempo. En inglés, la mayoría de eventos se corresponden con verbos, aunque a veces se pueden presentar eventos mediante sintagmas nominales (the move, the increase). Algunos verbos no representan eventos (took effect, make, take y have). Se utilizan enfoques de aprendizaje automático, basados tanto en reglas como en métodos estadísticos, para detectar eventos.
- *TimeBank*: corpus que consta de texto anotado procedente de artículos de noticias seleccionados de varias fuentes, incluyendo las colecciones Penn TreeBank y PropBank. Las expresiones temporales y eventos en este corpus están anotadas con TimeML, que proporciona enlaces temporales entre los eventos y las expresiones temporales que especifican la naturaleza de la relación (A abarca B, A ocurre antes que B, A y B son simultáneas, B abarca a A).

16. Question Answering (Búsqueda de Respuestas)

Consiste en la búsqueda de una información concreta en lugar de un documento completo. Está basado en técnicas de recuperación de información para encontrar los fragmentos con la información buscada. Los componentes de un sistema de *Question-Answering* son:

- Análisis de la pregunta: se debe saber el tipo de pregunta y el tipo de respuesta esperado. El tipo de pregunta determina si se trata de localizar un hecho concreto (pregunta factual: ¿Quién ganó el nobel de la paz en 2010?), si se busca la definición de un concepto (pregunta de definición: ¿Qué es un ipad?), o si se trata de una pregunta de tipo lista (¿Qué países se enfrentaron en la segunda guerra mundial?). En cuanto al tipo de respuesta, se suele indicar si se busca un nombre de persona, una organización, una fecha, un lugar, etc.
- Recuperación de información o documentos que pueden contener la respuesta: selecciona un número reducido de documentos de la colección o de internet.
- Selección de pasajes: los documentos seleccionados se analizan para escoger aquellas frases o fragmentos que pueden contener la respuesta a la pregunta realizada (entidades nombradas, palabras clave de la pregunta, proximidad de las

palabras clave, etc.). Los pasajes seleccionados serán la entrada del siguiente componente.

- Extracción de respuestas: procesa los pasajes o fragmentos de texto seleccionados para localizar y extraer la respuesta buscada. Se utilizan algoritmos basados en la extracción de patrones del tipo de respuesta que se espera (nombre de persona, año de nacimiento, definición, etc.) y basados en *N*-gramas (asignan un peso a cada unígrafo, bigrafo y trígrafo que aparece en el fragmento seleccionado).

17. Sistemas de extracción de resúmenes

La extracción de resúmenes es el proceso de extraer la información más importante de un texto para producir una versión reducida del mismo. Los tipos de resúmenes incluyen los siguientes:

- Síntesis de cualquier documento.
- Artículo científico.
- Titulares de noticias.
- Fragmento que resume cada resultado de un motor de búsqueda.
- Actas, líneas de acción o resúmenes de reuniones de negocios.
- Resúmenes de los temas tratados en los correos electrónicos.
- Frases comprimidas para producir textos simplificados.
- Respuestas a preguntas complejas, construidas resumiendo varios documentos.

A su vez, estos tipos de resúmenes se pueden clasificar en:

- Resumen de documentos simples: para producir titulares o síntesis.
- Resumen de documentos múltiples: para sintetizar noticias sobre un mismo evento o contenido Web sobre el mismo tema.
- Resumen genérico: no se considera ningún tipo de usuario en particular ni se tiene en cuenta una necesidad determinada de información. Lo importante es la información en sí.
- Resumen centrado en la consulta: se produce en respuesta a una pregunta del usuario y trata de satisfacer su necesidad de información.

También se puede distinguir entre la generación de un resumen o un extracto del texto. El extracto es más simple, y se construye combinando frases seleccionadas (extraídas) del documento que se está resumiendo. El resumen utiliza distintas palabras para describir los contenidos del documento.

17.1 Componentes de un algoritmo de resumen

17.1.1 Resúmenes de documentos simples

- Selección del contenido, que puede ser:
 - No supervisado: es básicamente una tarea de clasificación, que consiste en etiquetar cada frase del documento como *importante* o *no importante*. Para ello se seleccionan las frases con más palabras destacadas o informativas, utilizando el esquema *tf-idf*. De esta forma se mide la relevancia de una palabra teniendo en cuenta tanto la frecuencia como la inversa de la frecuencia en la colección, para controlar la aparición de palabras comunes pero no específicas.

$$peso(w_i) = tf_{i,j} \times idf_i$$

También se puede utilizar la relación *log-likelihood*, que funciona mejor que la anterior. Calcula la relación que existe entre la probabilidad de observar la palabra w , tanto en el corpus de entrada como en el de base (suponiendo las mismas probabilidades en ambos) y la probabilidad de observar dicha palabra w asumiendo distintas probabilidades de que aparezca en el corpus de entrada y en el de base. La fórmula para calcular el peso de cada palabra es:

$$peso(w_i) = \begin{cases} 1 & si -\log(\lambda(w_i)) > 10 \\ 0 & en cualquier otro caso \end{cases}$$

y el de cada frase:

$$peso(S_i) = \sum_{w \in S_i} \frac{peso(w)}{|\{w | w \in S_i\}|}$$

El algoritmo calcula este peso para cada frase y las clasifica por su puntuación. El resumen contiene las frases mejor clasificadas.

Se pueden tener en cuenta las relaciones de coherencia del discurso mediante las relaciones RST (Rhetorical Structure Theory), en las que las frases *núcleo* tienen más posibilidades de ser seleccionadas para realizar el resumen.

- Supervisado: se necesita un corpus de entrenamiento de documentos emparejados con extractos de resúmenes creados por personas. Se debe etiquetar cada frase del documento con un 1 si aparece en el resumen y un 0 si no está. El clasificador necesita elegir características para determinar si una frase debe aparecer en el resumen. Estas características son: posición (el título en los

artículos periodísticos, las primeras frases de cada párrafo, etc.), frases de referencia (in summary, in conclusion, this paper, para concluir, en resumen, etc.), palabras con contenido informativo, longitud de la frase (las cortas no suelen servir para un resumen), cohesión. Con estos datos se puede entrenar al clasificador para calcular las etiquetas (0 ó 1) para nuevas frases.

Si se va a realizar un resumen empleando frases que no son extraídas del texto hay que alinear cada documento con su resumen antes de entrenar el corpus. Así se podrán encontrar las frases del documento original que se han utilizado en el resumen, aunque se utilicen otras palabras. Los algoritmos de alineamiento permiten que los métodos supervisados para seleccionar el contenido puedan utilizar corpora paralelos.

- Ordenación de la información: se elige el orden en el que aparecen las frases en el documento.
- Realización de la frase: se corrigen las frases eliminando partes que no son esenciales, combinando varias frases en una sola o resolviendo problemas de coherencia. Las frases se pueden simplificar eliminando aposiciones, sintagmas preposicionales sin entidades nombradas, adverbios al principio de la frase, etc. o mediante la aplicación de técnicas de aprendizaje automático junto con un corpus paralelo de documentos con sus resúmenes.

17.1.2 Resúmenes de documentos múltiples

- Selección del contenido: cuando se realizan resúmenes a partir de varios documentos relacionados con un tema, se suelen utilizar métodos no supervisados, ya que no existen muchos corpus de entrenamiento para métodos supervisados. La diferencia que existe entre resumir un documento simple o múltiples documentos radica en la gran cantidad de redundancia que se produce en el contenido de los distintos textos seleccionados. Se puede incluir un factor de redundancia, basado en la similitud de sentencias, para elegir las frases que se van a extraer de los documentos en la realización del resumen, eliminando las que son similares a las que ya existen en el resumen. Otra opción es utilizar un algoritmo de agrupamiento sobre todas las frases de todos los documentos. De esta forma se crean grupos de frases relacionadas y se selecciona una sola frase central para cada grupo.
- Ordenación de la información: si se utiliza el orden cronológico del argumento del texto se pueden producir resúmenes sin cohesión. Esto se evita cogiendo fragmentos

de texto en lugar de frases sueltas. También es importante conservar la coherencia del discurso y tener cuidado con las correferencias.

- Realización de la frase: en esta etapa el principal problema son las correferencias que pueden dar lugar a falta de coherencia en el resumen. Para evitarlo, se pueden utilizar algoritmos de resolución de correferencias en la salida para extraer nombres, y establecer reglas que solucionen este inconveniente. En esta fase se emplean algoritmos de fusión de frases para combinarlas, alineando los documentos para encontrar frases similares.

17.2 Evaluación de los sistemas de resúmenes

La evaluación puede estar basada en las tareas (especificada en el punto 3.1) o ser independiente de ellas. En este último caso, la métrica que más se utiliza se llama *ROUGE* (Recall-Oriented Understudy for Gisting Evaluation). Existen versiones para unigramas, bigramas, etc., de tal forma que, por ejemplo, para medir la exhaustividad de bigramas se utiliza la fórmula:

ROUGE2

$$= \frac{\sum_{S \in \{Resúmenes\} de referencia(humano)} \sum_{bigramas \in S} NumCocincidencias(bigramas)}{\sum_{S \in \{Resúmenes\} de referencia} \sum_{bigramas \in S} Num(bigramas)}$$

La función *NumCocincidencias(bigramas)* devuelve el número máximo de bigramas que aparecen tanto en el resumen automático como en el realizado por una persona.

ROUGE mide la exhaustividad, pero está inspirada en otra métrica, llamada *BLUE*, que se basa en la precisión y que se utiliza para evaluar los resultados de los sistemas de traducción automática. La fórmula es similar a la anterior, pero el denominador es la suma total del número de bigramas que existen en los resúmenes automáticos. Por lo tanto, *ROUGE* determina cuantos bigramas de los resúmenes hechos por humanos están cubiertos por los que aparecen en el resumen candidato, mientras *BLEU* calcula cuantos bigramas de la traducción candidata aparecieron en el resumen de referencia hecho por personas, llamado también resumen modelo.

Algunas variantes de *ROUGE* son: *ROUGE-L*, que mide la secuencia común más larga en los resúmenes modelo y los resúmenes candidatos o automáticos; *ROUGE-S* y *ROUGE-SU*, que calculan el número de *skip-bigrams*, que son pares de palabras (bigramas) entre los que puede haber otras palabras incluidas. Pero la gran diferencia

que existe entre los resúmenes modelo y los automáticos, hace que la coincidencia sea muy baja y que *ROUGE* no sea una métrica válida.

Este problema determina el uso de otras métricas que se centran más en el significado. Por ejemplo, el *Método Pirámide* cuenta el número de unidades de significado (Summary Content Units, SCU) compartidas entre los resúmenes modelo y los candidatos, valorando también la importancia de dichas unidades (mayor peso cuantos más resúmenes modelo las utilicen). Se construye una pirámide de SCUs, colocando en la parte superior las que tienen mayor importancia. El peso de cada resumen se calcula con la ecuación:

$$D = \sum_{i=1}^n i \times D_i$$

dónde D_i es el número de SCUs de un resumen que aparecen en cada nivel de la pirámide. El mejor resumen será aquel que contenga más SCUs de los niveles superiores.

18. Conclusiones

El lenguaje y la comunicación constan de un conjunto de frases que tienen una estructura y que se relacionan de forma coherente, constituyendo una organización con significado llamada discurso. La desambiguación del contenido de estas frases se basa en determinar las relaciones de referencia y coherencia, tanto del significado como de los hechos que se relatan en el discurso. Detectar estas relaciones de coherencia en la clasificación automática de textos es importante para establecer su calidad. El contenido del discurso se puede utilizar posteriormente para hacer resúmenes o resolver consultas. En los sistemas de extracción de resúmenes y de búsqueda de respuestas suele haber un paso previo en el que se detectan y clasifican las entidades nombradas que aparecen en los documentos que se van a utilizar: personas, lugares, organizaciones, términos específicos de un dominio, fechas, horas, eventos, precios, etc. La detección de estas entidades y el establecimiento de un orden en los eventos y expresiones temporales facilitan la realización de resúmenes coherentes y la resolución de consultas o búsqueda de información. Por lo tanto, se deben localizar eventos que ocurren antes, después o durante determinado periodo de tiempo o hecho concreto.

Los pasos que se deben llevar a cabo en la realización de resúmenes son: selección del contenido, ordenación de la información y realización de la frase. Si se realizan a partir de varios documentos se deben evitar tanto la redundancia en el contenido como la falta

de coherencia que se puede producir por las correferencias. La evaluación de los sistemas de búsqueda de respuestas y de extracción de resúmenes determinará la calidad de los resultados obtenidos, su exhaustividad y su precisión.

Bibliografía

JURAFSKY, D., et al. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. 2st ed. Upper Saddle River: Prentice Hall, 2008. pp. 1-157, 385-639, 681-810. ISBN-13: 978-0131873216

Referencias

- [1] ANAND KUMAR, M., et al. “A Sequence Labeling Approach to Morphological Analyzer for Tamil Language”. *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, nº 06 (2010), p. 194-195. Disponible en: <http://www.enggjournals.com/ijcse/doc/IJCSE10-02-06-96.pdf>
- [2] BILGIN, A.; ELLSON, J.; GANSNER, E.; HU, Y.; NORTH, S. *Graphviz-Graph Visualization Software* [programa]. Ver. 2.34. AT & T Labs Research, 2013. Disponible en: <http://www.graphviz.org/>
- [3] CHAKRABARTI, S. *Mining the Web. Discovering Knowledge from Hypertext Data*. 1st ed. San Francisco: Morgan Kaufman, 2003. pp. 4-6. ISBN: 1558607544.
- [4] COOPER, R. *Quantification and syntactic theory*. Dordrecht: Springer Netherlands, 1983. pp. 52-78.
ISBN: 978-94-015-6932-3
- [5] DepPattern [programa]. *Grupo ProLNat. Universidad de Santiago de Compostela*. Disponible en: <http://gramatica.usc.es/pln/tools/deppattern.html>
- [6] GONZALEZ-AGIRRE, A., LAPARRA, E., RIGAU, G. “Multilingual Central Repository version 3.0: upgrading a very large lexical knowledge base”. En *GWC 2012 6th International Global Wordnet Conference*. pp. 118.
- [7] HROMKOVIČ, J.; SEIBERT, S.; WILKE, T. “Translating Regular Expressions into Small ϵ -Free Nondeterministic Finite Automata”. *Journal of Computer and System Sciences*, vol. 62, nº 4 (2001), pp. 565-588. Disponible en: <http://www.sciencedirect.com/science/article/pii/S0022000001917489>
- [8] KARTTUNEN, L.; KAPLAN, R. M.; ZAENEN, A. “Two-level morphology with composition”. En *Proceedings of the 14th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1992. pp. 141-148. Disponible en: <http://acl.ldc.upenn.edu/C/C92/C92-1025.pdf>
- [9] LEVITHAN, S. *Regexpal 0.1.4 – a JavaScript regular expression tester*. 2012. Disponible en: <http://regexpal.com/>
- [10] MOHRI, M. “Finite-state transducers in language and speech processing”. *Computational linguistics*, vol. 23, nº 2 (1997), pp. 269-311. Disponible en: <http://acl.ldc.upenn.edu/J/J97/J97-2003.pdf>

- [11] MOHRI, M.; PEREIRA, F.; RILEY, M. “Weighted finite-state transducers in speech recognition”. *Computer Speech & Language*, vol. 16, nº 1 (2002), pp. 69-88. Disponible en: <http://www.cs.nyu.edu/~mohri/pub/csl01.pdf>
- [12] PADRÓ, L. *FreeLing. An open source suite of language analyzers*. [programa en línea]. 2012. Versión 3.1. Disponible bajo licencia GNU/GPL en: <http://nlp.lsi.upc.edu/freeling/demo/demo.php>
- [13] PORTER, M. *Snowball Stemmer Demo*. [programa en línea]. 2012. Disponible en: <http://snowball.tartarus.org/>
- [14] REICHENBACH, H. *Elements of symbolic logic*. Dover Publications, 1980. ISBN-10: 0486240045
- [15] ROCHE, E.; SCHABES, Y. “Deterministic part-of-speech tagging with finite-state transducers”. *Computational linguistics*, vol. 21, nº 2, (1995), pp. 227-253. Disponible en: <http://acl.ldc.upenn.edu/J/J95/J95-2004.pdf>
- [16] SANTANA, O. *Flexionador y lematizador de palabras del español* [programa en línea]. Grupo de Estructuras de Datos y Lingüística Computacional de la Universidad de Las Palmas de Gran Canaria. 2007. Disponible en: <http://protos.dis.ulpgc.es/investigacion/scogeme02/lematiza.htm>.
- [17] SHIEBER, S. M. *An introduction to unification-based approaches to grammar*. Stanford, CA: Center for the Study of Language and Information, Stanford University, 1986. pp 24-36.
ISBN-10: 0226122158
- [18] VENDLER, Z. *Linguistics in philosophy*. Ithaca: Cornell University Press, 1967. pp. 97-146.
ISBN-10: 0801404363