Introducción histórica y tecnológica

Indice

Introducción histórica y tecnológica

I. Introducción

¿Cómo surgió el NLP? ¿En qué consisten las técnicas fundamentales? ¿Qué es una cadena NLP? A lo largo de esta unidad, se responderá a todas estas preguntas. El NLP es una rama multidisciplinar muy extensa y, por tanto, aunque muchas de las cosas explicadas en esta unidad no serán puestas en práctica en este módulo, es interesante conocerlas para poder profundizar en ellas en un futuro, si se desea.

Hoy en día, en parte gracias a los últimos avances en lo que se conoce como Deep Learning, el procesamiento de lenguaje natural ha avanzado mucho. Sin embargo, es interesante conocer los orígenes y las técnicas clásicas, ya que son el fundamento en el cual radican estas nuevas técnicas, que exceden el objetivo de una asignatura introductoria como esta.

II. Objetivos

El objetivo de esta unidad es dar a los alumnos unas nociones teóricas básicas. No es, por tanto, necesario profundizar en todos los conceptos, que se ven muy por encima, ni se debe emplear demasiado tiempo en intentar comprender todos los detalles. Naturalmente, quien tenga interés en profundizar, una vez acabado el módulo, puede leer el libro recomendado en la bibliografía, donde se desarrollan todos estos puntos.

Este documento pretende servir de guía puramente teórica para que los alumnos conozcan las bases en las que se cimenta el procesamiento del lenguaje natural. Así pues, no tiene una relación directa con ninguna de las actividades prácticas que se expondrán a lo largo de la asignatura, sino que simplemente servirá como una ambientación inicial, que sin duda facilitará la comprensión y el aprendizaje del resto de actividades.

Bastará, por tanto, con realizar una lectura detenida de este documento y con contestar a un breve cuestionario que se propone como ejercicio.

III. Contexto histórico

El **procesamiento del lenguaje natural** (PLN o NLP por sus siglas en inglés: Natural Language Processing) es una rama de la Inteligencia Artificial que aúna el conocimiento de dos disciplinas: la computación y la lingüística. Por tanto, abarca todas las técnicas que tratan de emular nuestras capacidades innatas para entender y transmitir el lenguaje humano.

El origen del NLP tiene lugar en los años 50, cuando Turing define su famoso test (“Test de Turing”), el cual determina si una máquina es realmente inteligente o no, vinculando esa inteligencia a la capacidad para responder preguntas sin que un humano perciba que se trata de un sistema artificial. Durante esa década, universidades de Estados Unidos y Reino Unido se involucran en importantes proyectos de traducción automática. Es una época de gran optimismo, en la que todo parecía posible y abarcable en un periodo muy corto de tiempo —suponían que en 3 o 5 años serían capaces de traducir correctamente cualquier texto—.

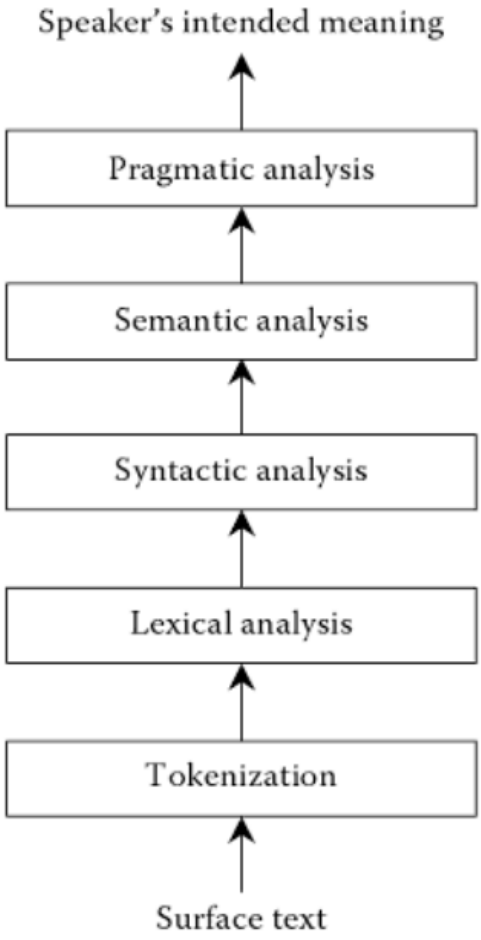
A partir de los años 60, el optimismo y las inversiones decrecieron por la falta de resultados prácticos. Hasta que en los 80, con el surgimiento de nuevas técnicas estadísticas comenzaron a surgir nuevas formas de enfrentarse a los mismos problemas de siempre. En lugar de emplear una aproximación simbólica —complejas y numerosas reglas escritas manualmente— se dio paso a una aproximación empírica, basada en grandes corpus textuales con los cuales los sistemas eran entrenados y aprendían por sí solos.

> En la actualidad, ambas técnicas conviven y tienen cabida. Para resolver problemas del mundo real, se utilizan métodos híbridos que integran lo mejor de ambas aproximaciones.

IV. Cadenas de procesamiento clásicas

Un elemento fundamental, que se verá constantemente y directamente en este documento e implícitamente en todas las prácticas de la asignatura, son las denominadas cadenas de procesamiento. Cada vez que se resuelva un problema NLP, se irán enlazando unos procesos con otros, de forma que la salida de un proceso se convierta en la entrada del siguiente, y así sucesivamente. Esta forma de resolver las cosas es un estándar en el mundo NLP, ya que se ha comprobado ampliamente su efectividad. Consiste en dividir un problema en subproblemas secuenciales, donde cada fase sucesiva de nuestra cadena resolverá únicamente un problema específico.

Figura 1.1. Fases de análisis en el PLN.



Fuente: Handbook of NLP(Chapman & Hall).

Como se muestra en la imagen superior, la entrada al sistema es el texto por analizar, que va pasando por procesos sucesivos hasta que se obtiene el significado del mensaje. Se trata de un modelo clásico o tradicional, ya que hoy en día existen otras técnicas alternativas que pueden variar algo este esquema. Pero sigue siendo un modelo vigente, que, además, nos servirá para comprender de forma teórica en qué consiste el NLP.

Desde el punto de vista teórico, las fases del análisis lingüístico son:

Sintaxis

Obtención de la estructura de las frases.

Semántica

Obtención del significado literal de las frases.

Pragmática

Obtención del significado en contexto.

Naturalmente, tal y como se ha dicho previamente, esto no es más que un modelo teórico muy usado durante la etapa simbólica, pero que aún mantiene su vigencia. Simplemente, en la práctica, a la hora de implementar estas teorías, se ha comprobado que no es tan fácil realizar esta separación. La realidad es que entre una etapa y otra los límites son muy borrosos y, en las fases superiores (semántica y pragmática), las dificultades son tan numerosas como variadas las formas de afrontarlas.

¿En qué consisten estos procesos sucesivos? Para obtener los árboles y estructuras sintácticas, previamente habrá que localizar las palabras y frases (tokenización) y haber asignado, mediante el análisis léxico, a cada palabra su correspondiente etiqueta morfológica (verbo, adjetivo...) y su lema. El lema es la raíz de la palabra (comeríamos->comer, perritas->perro), algo así como la entrada de un diccionario.

Pero esto no es todo. El PLN es una disciplina que está formada por dos áreas de conocimiento contrapuestas y complementarias:

Análisis del Lenguaje Natural

Conocido por sus siglas en inglés: NLA (Natural Language Analisys). Corresponde a las cadenas de procesamiento que se acaban de explicar, es la parte del NLP que pretende analizar un texto para llegar a extraer su significado de forma automática.

Generación del Lenguaje Natural

En inglés, NLG (Natural Language Generation). Son todas aquellas técnicas que tratan de generar frases a partir de un conocimiento del sistema; es decir, que emulan la capacidad opuesta al NLA. No todos los sistemas que responden a preguntas son de tipo NLG, lo son aquellos más avanzados que no devuelven una respuesta “enlatada” y poco natural extraída de una base de datos de respuestas. La generan en tiempo real, partiendo de una información previa del sistema y de un modelo de generación del lenguaje aplicado a expresar dicha información.

A continuación, en los siguientes apartados de este documento, se profundizará en cada una de las fases que componen las cadenas clásicas del PLN.

V. Preprocesamiento de textos

El **preprocesamiento de textos** es la primera de las fases y puede subdividirse a su vez en dos subfases: limpieza y segmentación de textos.

La limpieza de textos (en inglés, Document Triage) es un proceso en el que se obtienen unos ficheros digitales y se convierten en documentos de texto perfectamente definidos. El objetivo es la obtención de grandes corpora textuales. Un corpus no es más que una colección de textos, que en las aproximaciones estadísticas se usa masivamente para entrenar sistemas NLP. Hoy en día es común el uso de arañas y programas de scraping para descargar de Internet estos corpus, pero en sus inicios debía realizarse mucho trabajo de recopilación manual.

El primer problema al que hay que enfrentarse aquí es el de la codificación de caracteres. Hay diversos códigos y cada uno de ellos tiene distinto número de bits:

- ASCII: 7 bits, permite 128 caracteres.
- ISO-8859: 8 bits, 256 caracteres.
- Big-5, GB: para el chino, con 16 bits, 65.536 caracteres.
- Unicode 5.0, UTF-8: tiene entre 1-4 bytes.

Posteriormente, hay que proceder a la identificación del idioma. Si a nuestro sistema entran muchos textos y muy diversos, habrá que clasificarlos por idioma para poder escoger el tipo de analizador que debe aplicarse en fases posteriores —no es igual un parser sintáctico para alemán que para español o chino—. Se trata de un problema que podría parecer fácil a simple vista. De hecho, existen bastantes librerías que resuelven con gran precisión esta tarea. Para ello, asumen que algunos caracteres solo existen en ciertos idiomas y además aplican algoritmos de distancia vectorial que tienen en cuenta cuáles son los caracteres más frecuentes y en qué orden se dan, incluso sin requerir el uso de diccionarios multilingües, que harían el proceso mucho más costoso computacionalmente.

Pero como siempre, y como se irá viendo a lo largo de todas las fases de la cadena NLP, una cosa es la teoría y otra la práctica. Para textos de longitud muy corta o para documentos multilingües, estos algoritmos no funcionan tan bien, con lo cual, algo tan en boga hoy en día como el análisis automático de tweets puede convertirse en una tarea más compleja de lo esperado.

A continuación, una vez detectado el idioma, se procede a la segmentación de textos, que tal y como se verá en los siguientes apartados del presente documento se compone de la fase de tokenización y la fase de análisis léxico.

VI. Tokenización

La tokenización no es más que la segmentación de un texto en palabras. O si se es más preciso, se trata de la separación en tokens. Un token es la unidad lingüística mínima usada en NLP, que no tiene por qué ser siempre exactamente lo mismo que una palabra.

Se analizará la siguiente frase: 'hacedme partícipes del éxito'. ¿Cuáles serían sus tokens constitutivos?: 'haced', 'me', 'partícipes', 'de', 'el', 'éxito', '.'. Obsérvese que se han distinguido más tokens que palabras (haced+me, de+el) y que además se ha considerado el punto final como un token independiente.

Esto, que a simple vista puede parecer relativamente sencillo de programar para idiomas como el castellano, puede llegar a complicarse mucho para idiomas no segmentados (chino, japonés, tailandés...). Las palabras en la mayor parte de los idiomas europeos están separadas por espacios, pero en idiomas orientales las palabras se escriben una detrás de otra sin ningún tipo de separación.

Para ambos tipos de idiomas existen además distintas formas de división interna dentro de cada palabra. Relacionándolo con esta idea, se distinguen distintos tipos de morfología de las palabras:

Aislante

Las palabras no se dividen en unidades más pequeñas, sino que suelen constar de un único morfema (por ejemplo, chino mandarín).

Sintética

Las palabras sí se dividen en unidades más pequeñas, denominadas morfemas, siendo por tanto raramente monomorfémicas. Suelen tener una raíz y algunos morfemas.

- **Aglutinante**: subtipo de morfología sintética donde además los morfemas aparecen claramente divididos (japonés).
- **Inflexional o fusional**: morfología sintética donde los límites entre morfemas no son claros (latín). Tiene que ver con las varias formas de una palabra dentro del mismo paradigma, como habl-o, habl-as, habl-a, habl-amos, habl-áis, habl-an, habl-é, habl-aste, habl-ó, etc. No es posible separar un juego de morfemas que indique solamente número (singular o plural) y otro juego que indique solamente el caso (nominativo, acusativo, etc.), como probablemente se podría hacer en un idioma aglutinante.

Polisintética

Caso extremo de morfología aglutinante, donde los morfemas se unen para formar complejas palabras que llegan a funcionar incluso como una frase completa (chukchi).

VII. Segmentación de frases

Dividir un texto en frases es un problema relativamente sencillo en idiomas con puntuación como el español. Sin embargo, aun así existen ciertas dificultades, por ejemplo, con abreviaturas y siglas.

Esto se puede apreciar en el siguiente ejemplo: “El sr. P. Gonzales nació en U.S.A. mide 3.5 m y su dominio es www.pgonzalez.com”. Y si se consideran textos de Internet o mensajes de móvil donde el usuario no se molesta en puntuar ni iniciar con mayúscula la primera palabra de la frase, de nuevo, se halla que la realidad plantea muchos más problemas que la teoría.

En idiomas sin puntuación —por ejemplo, el tailandés—, resulta bastante compleja la separación, lo que constituye un problema similar al de la delimitación de palabras sin separación.

VIII. Análisis léxico o morfológico

Una vez obtenidos los tokens, hay que asignar a cada uno de ellos un lema y una etiqueta morfológica. A este proceso se le denomina análisis léxico o análisis morfológico.

Una palabra es el conjunto de uno o varios morfemas. Estos morfemas pueden ser el lexema (la raíz o morfema principal) y los morfemas derivativos (con menor relevancia que el lexema, que según su colocación pueden ser prefijos, infijos o sufijos).

Caserón→ Cas (lexema) - er (infijo) - on (sufijo).

La lematización es el proceso consistente en obtener el lema de una palabra (por ejemplo: comeríamos→ comer, perritas→ perro). Un proceso muy similar, aunque algo más sencillo y liviano computacionalmente, es la obtención de la raíz (proceso denominado stemming, en inglés). No es tan preciso como la lematización (comeríamos→ com, perritas→ per), ya que puede dar lugar a confusión con otras palabras de distinto lema, pero misma raíz (peras→ per). Para el español es más adecuado aplicar la lematización, mientras que para el inglés el stemming podría ser suficiente, ya que es un idioma inflexionalmente menos complejo y por tanto con menos conflictos entre raíces. Todo depende del grado de precisión que requiera nuestro sistema.

Además, en la fase de lematización suelen normalizarse los textos a una forma canónica (1→ uno, una→ uno, unos→ uno).

Las etiquetas morfológicas, también denominadas etiquetas POS (del inglés, Part of Speech), indican la categoría morfológica del token analizado (verbo, adjetivo…). Existen distintos estándares internacionales, como el EAGLES o el de Penn Tree Bank, que uniformizan la forma que adoptan dichas etiquetas.

Por ejemplo, la lista completa para Penn Tree Bank es la siguiente:

Tabla 1.1. Lista completa para el Penn Tree Bank.

Number	Tag	Description

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer

17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present

33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

En todas las fases de la cadena de NLP, los algoritmos deben afrontar el mayor de los problemas: la ambigüedad. Existen muchas posibilidades y hay que escoger una. En el caso del análisis morfológico, hay múltiples etiquetas posibles, pero solo una válida. Por ejemplo, 'perro' puede ser sustantivo (“El perro come zanahorias”) o adjetivo (“He tenido un día muy perro”) y la máquina tendrá que elegir cuál de las dos posibilidades es la adecuada.

¿Cómo funcionan los algoritmos de desambiguación morfológica?

Algunos usan extensos diccionarios con todas las palabras flexionadas —por ejemplo, el verbo 'comer' y todas sus formas: 'como', 'comes', 'comeríamos', 'comeré'…—. Otros almacenan un pequeño diccionario solo con los lemas y además usan reglas de combinación para generar, solo cuando sea necesario, todas las formas posibles por cada lema. Estas reglas sirven tanto para la generación morfológica (generar una forma a partir de su lema y sus rasgos morfológicos: comer+1ªpersonaPluralPresenteIndicativo → comemos) como para la lematización (comemos → comer). Además del análisis morfológico, otras disciplinas de PLN se benefician de estas técnicas, por ejemplo, la traducción automática, una vez que detecta el lema y sus rasgos morfológicos en el idioma origen, deberá generar el lema traducido en el idioma destino conservando dichos rasgos. Por tanto, lematización y generación morfológica son dos procesos de sentido opuesto.

Transductores de Estados Finitos

Uno de los modelos más usados para formalizar estos algoritmos es el de los Transductores de Estados Finitos (en inglés, FSTs: Finite State Transducers). Son un tipo de máquina/autómata de Estados Finitos (FSM: Finite State Machine) con dos cintas: una de entrada y otra de salida. En los autómatas existen una serie de estados (las bolitas del diagrama) y de transiciones (las flechas) entre esos estados. De una forma simplificada, estas máquinas tienen una cinta de entrada por la que les van llegando sucesivos caracteres (primero la 'g', luego la 'l', después la 'a'…) y cada vez que les llega un carácter de entrada generan otro de salida (si entra una 'g' generan una 'g'). De esta forma, la máquina está programada para que, si se encuentra en el estado 1 y le llega una 'g', genera una 'g' y pasa al estado 2, y así sucesivamente. Si llega al estado 6 y le entra un ^ (símbolo que representa un divisor entre el lexema y el morfema derivativo), entonces saca por la cinta de salida un símbolo que representa un carácter vacío (equivalente a no sacar ningún carácter) y pasa al estado 7. Ya en el estado 7, sin necesidad de que le entre nada, genera una 'e' y pasa al estado 8. Si en el estado 8 le entra una 's', genera una 's' y finaliza en el estado 9, que es un estado final.

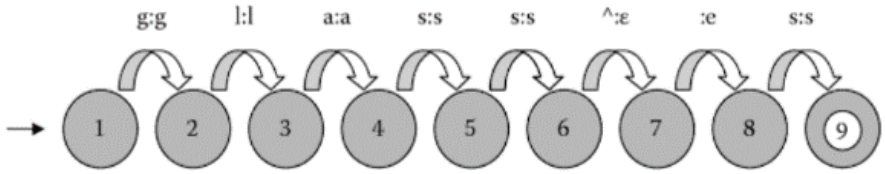


FIGURE 3.1 A spelling rule FST for *glasses*.

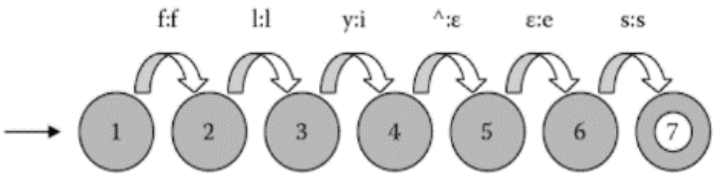


FIGURE 3.2 A spelling rule FST for *flies*.

Figura 1.2. FST para 'glasses' y 'flies'.
Fuente: Handbook of NLP (Chapman & Hall).

Así, el primer autómata de la figura modeliza una regla para obtener o detectar el plural de la palabra 'glass' ('glass' → 'glasses'), mientras que el segundo obtiene o detecta el plural de 'fly' ('fly' → 'flies'). Por tanto, los FSTs sirven tanto para la generación morfológica ('fly' + 's' = flies') como para la lematización ('flies' → 'fly').

Anotación: DATR

Existen lenguajes como DATR (véase el ejemplo de la figura 1.3.) que modelizan estos FSTs. Así, la primera regla de generación morfológica dice que, si se desea generar 'glass' sin ningún morfema, hay que escribir 'glass' en la cinta de salida, y la segunda dice que, si se desea generar 'glass' con morfema de plural 's', habrá que escribir 'glasses'.

```
<g l a s s #> = g l a s s.
<g l a s s ^ s #> = g l a s s e s.
<f o x #> = f o x.
<f o x ^ s> = f o x e s.
<c a t #> = c a t.
<c a t ^ s #> = c a t s.
```

Figura 1.3. DART para 'glass', 'fox' y 'cat'. *Fuente: Handbook of NLP* (Chapman & Hall).

Lo anteriormente explicado (FSTs) es simplemente una implementación del primero de los tres tipos de modelos de análisis léxico que existen, que son distintas aproximaciones a la estructura interna de las palabras:

Unidad y disposición

En inglés, I&A: Item and Arrangement: está vinculada a la denominada Morfología de Estados Finitos (Finite State Morphology). Según este modelo, el análisis de la información de una palabra (looked) es la suma de la información de su lexema (look) más la de sus morfemas derivativos (ed). Esto se modeliza muy bien con Transductores de Estados Finitos.

Unidad y proceso

I&P, Item and Process: está vinculada a la denominada Morfonología de Estados Finitos (Finite State Morphonology). Es similar al anterior, pero tiene la capacidad de modelizar palabras que se transforman de una forma más compleja (sing → sang, fox → foxes, en lugar de sing → singed, fox → foxs). Tiene en cuenta los criterios fonológicos, no solo morfológicos —de ahí la palabra morfonología = morfología + fonología— y por tanto debe lidiar con las variantes de pronunciación y su correspondiente escritura ortográfica.

Palabra y paradigma

W&P, Word and Paradigm: cada lema está asociado con una tabla o paradigma que relaciona cada una de las variantes de dicho lema con sus correspondientes propiedades morfosintácticas. Idiomas como el ruso son más fáciles de modelizar mediante este paradigma.

IX. Análisis sintáctico

La sintaxis se define como la disciplina lingüística que estudia el orden y la relación de las palabras o sintagmas en la oración, así como las funciones que cumplen. Esta fase, a diferencia de la anterior (morfoanálisis) no intenta etiquetar cada palabra, sino que busca las relaciones entre ellas y sus distintas formas de agruparse.

Existen dos formas de representación de análisis sintácticos ampliamente extendidas:

- Los árboles sintácticos: las distintas categorías sintácticas constituyen los nodos de un árbol cuyos elementos inferiores —las hojas del árbol— son las propias palabras analizadas.
- La notación simplificada mediante paréntesis.

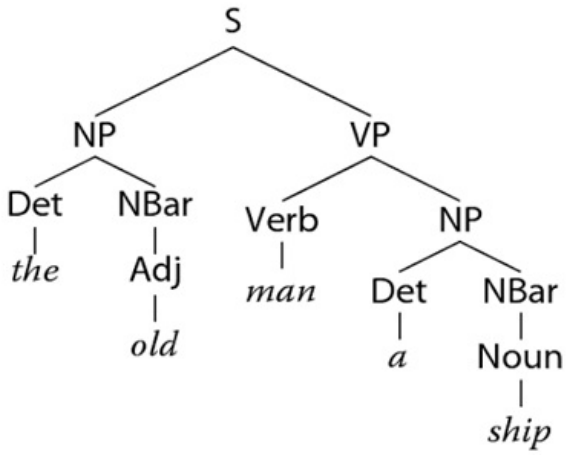


Figura 1.4. Análisis sintáctico de la frase 'the old man a ship'.
Fuente: Handbook of NLP (Chapman & Hall).

A medida que se avanza en las sucesivas etapas de la cadena NLP, el nivel de complejidad va aumentando, debido al ya mencionado problema de la ambigüedad. Si en las etapas anteriores ya existía esta complicación, será en esta fase cuando la explosión combinatoria plantee verdaderas dificultades y los algoritmos se optimicen al máximo para limitar el número de posibilidades que hay que explorar.

- Put the block [in the box on the table]

- Put [the block in the box] on the table

Figura 1.5. Ambigüedades sintácticas.
Fuente: Handbook of NLP **(Chapman & Hall).**

A diferencia de los lenguajes artificiales —como los lenguajes de programación: python, java…—, el lenguaje natural tiene infinitas posibilidades. Las gramáticas de los lenguajes de programación son no ambiguas y permiten el *parseado* en un tiempo linealmente proporcional a la longitud del texto de entrada. La capacidad generativa de los lenguajes artificiales, es decir, el poder de los formalismos gramaticales que usan, es mucho más reducido, suelen emplearse las denominadas Gramáticas Libres de Contexto. Sin embargo, el lenguaje humano no tiene límites y esto se aprecia, por ejemplo, en la cantidad infinitamente variable de información que se puede añadir entre dos elementos relacionados ('who' y 'to' en el siguiente ejemplo):

Who did you sell the car to __?

Who do you think that you sold the car to __?

Who do you think that he suspects that you sold the car to __?

Figura 1.6. Capacidad generativa.
***Fuente: Handbook of NLP*(Chapman & Hall).**

9.1. Gramáticas Libres de Contexto

Las gramáticas libres de contexto (en inglés, Context Free Grammars, CFGs) son el formalismo gramatical que más influencia ha tenido desde que fue introducido por Chomsky en 1956.

Anotación: Una gramática G está formada por:

- un conjunto de elementos terminales (T).
- un conjunto de elementos no terminales (N).
- un símbolo de inicio (S) perteneciente a N.
- un conjunto R de reglas de producción (A → B).

O más formalmente expresado: G={T,N,S,R}

A continuación, se muestra un ejemplo simplificado de gramática, para comprender estos conceptos:

NBar → Boun	Det → a | an |the
NBar → Noun	Adj → old
NBar → Adj	Noun → man | men | ship | ships
VP → Verb	Verb → man | mans
VP → Verb NP	

Figura 1.7. Gramática de ejemplo.
***Fuente: Handbook of NLP*(Chapman & Hall).**

Se trata de una CFG compuesta por 11 reglas de producción, donde los elementos terminales son las palabras 'a', 'an', 'the', 'old', 'man', 'men'… y los elementos no terminales son todos aquellos símbolos que están en la parte izquierda de las reglas. Nótese la ambigüedad de algunas palabras como 'man' que puede ser un sustantivo ('man'='hombre'), pero también un verbo ('man'='tripular').

Los símbolos no terminales que se acaban de ver son de dos tipos:

- Categorías gramaticales:
  - Det → determinante
  - Adj → adjetivo
  - Noun → nombre
  - Verb → verbo
- Categorías sintácticas (frases):
  - S → oración
  - NP → sintagma nominal (en inglés, Noun Phrase)
  - NBar → cláusula intermedia que sirve para formar los NP
  - VP → sintagma verbal (Verbal Phrase)

Así, según la primera regla, una frase se forma cuando nuestra gramática encuentra un NP seguido de un VP. Y la última regla define que el símbolo no terminal 'Verb' debe activarse al encontrar la palabra 'man' o la palabra 'mans'. Se trata de una forma simplificada (concretamente CNF relajada I) de CFG.

Existen distintas formas de limitar las gramáticas libres de contexto aplicando restricciones en la forma de expresarlas, que se denominan formas normales. Así se podrá transformar una CFG en otra gramática equivalente, pero en su forma normal, que es mucho más manejable para los algoritmos que deben lidiar con ellas.

Estas son las formas normales:

Forma normal de Chomsky

En inglés, Chomsky Normal Form, CNF: solo admite en la parte derecha de la regla un único terminal o bien dos no terminales. Se resuelve mediante el algoritmo CKY, que se describirá más adelante.

- A → w
- A → BC

CNF relajada I

Como la CNF, pero además admite un único no terminal.

- A → w
- A → BC
- A → B

CNF relajada II

Como la CNF, pero admite uno o más no terminales (CNF solo admite 2 y CNF relajada I admitía tanto 1 como 2)

- A → w
- A → B...

Sin embargo, en la práctica las CFG no son lo más usado para analizar el lenguaje natural, por su limitada expresividad y su dificultad de uso, sino que se suelen utilizar algunas extensiones o derivaciones de esta teoría.

9.1.1. Algoritmo CKY

El algoritmo CKY (Cocke Kasami Younger), descrito en los años 60, es una de las implementaciones más sencillas de las gramáticas libres de contexto. Solo funciona con gramáticas CNF y, por tanto, previamente tendrán que haber sido transformadas todas nuestras reglas al formato 'A → w' y al formato 'A → BC'. Cualquier otro formato, como el 'A → B', no es válido, por lo tanto, reglas como 'VP → Verb' no son válidas. Existen unas reglas para realizar dicha transformación, pero se obviarán en este documento, ya que excederían el alcance de este curso.

Aquí se pueden ver las reglas originales (en rojo se destacan aquellas que no están en CNF):

S → NP VP	Det → a | an | the
NP → Det NBar	Adj → old
Nbar → Adj Noun	Noun → man | men | ship | ships
Nbar → Noun	Verb → man | mans
Nbar → Adj	
VP → Verb	
VP → Verb NP	

Supongamos que ya las hemos pasado a la forma CNF:

S → NP VP	Det → a | an | the
NP → Det NBar	Adj → old
Nbar → Adj Noun	Noun → man | men | ship | ships
Nbar → man | men | ship | ships	Verb → man | mans
Nbar → old	
VP → man | mans	
VP → Verb NP	

A continuación, se mostrará cómo funciona este algoritmo en detalle, con la frase “the old man a ship” (“el viejo conduce un barco”):

Paso 1

En primer lugar, se comienza por la fila inferior de la tabla (llamémosla fila 0) y se recorre, una a una, todas las palabras de esta fila, buscando para cada una de ellas posibles reglas que se disparen:

- the solo dispara la regla 'Det → a | an | the', así que se anota en la segunda fila 'Det'.
- old dispara la regla 'Adj → old', así que se anota 'Adj'. Al mismo tiempo, el Adj que se acaba de generar dispara otra regla: 'Nbar → Adj', así que se anota también 'NBar'.
- man dispara 4 reglas, se anota Noun, Verb, Nbar y VP.
- a se anota 'Det'.
- ship se anota 'Noun'.

Det	Adj, NBar	Noun, Verb, Nbar, VP	Det	Noun, NBar
the	old	man	a	ship

S → NP VP
NP → Det NBar
Nbar → Adj Noun
Nbar → man | men | ship | ships
Nbar → old
VP → man | mans
VP → Verb NP

Det → a | an | the
Adj → old
Noun → man | men | ship | ships
Verb → man | mans

Lo que se ha hecho hasta ahora es anotar en la fila 1 todos los posibles análisis de las palabras de entrada. Se ha llamado a esta fila “fila 1” y no fila 2, que podría parecer más natural, al estar situada en segundo lugar contando desde abajo, precisamente para especificar que los símbolos terminales de entrada han sido recorridos de uno en uno.

Paso 2

A continuación, se probará a formar combinaciones de 2 símbolos terminales de entrada (2 palabras sucesivas de la frase). Deben buscarse, por tanto, etiquetas morfosintácticas que puedan aplicarse a 'the old', 'old man', 'man a', 'a ship'. Esto se podrá hacer buscando combinaciones de 2 símbolos sucesivos de la fila 1 que activen nuevas reglas:

- the old = Det [Adj,Nbar]: estas 2 columnas admiten 2 combinaciones ('Det-Adj' y 'Det-NBar'). 'Det Adj' no hace saltar ninguna regla y 'Det-NBar' sí que activa la regla 'NP → Det NBar', con lo cual se genera en la fila 2 la anotación 'NP'.
- old man = [Adj,NBar] [Noun,Verb,NBar,VP]: estas 2 columnas generan 8 posibles combinaciones: 'Adj-Noun', 'Adj-Verb', 'Adj-NBar', 'Adj-VP', 'NBar-Noun', 'NBar-Verb', 'NBar-NBar', 'NBar-VP'. Y solo la primera de ellas (Adj-Noun) dispara una regla: Nbar → Adj Noun. Se anota, por tanto, NBar.

- man a = [Noun,Verb,NBar,VP] [Det]: 4 posibles combinaciones: 'Noun-Det', 'Verb-Det', 'NBar-Det', 'VP-Det'. Ninguna de ellas dispara una regla. No se anota, por tanto, nada.
- a ship = Det Noun: 1 posible combinación: 'Det-Noun'. Dispara la regla 'NP → Det NBar'. No se anota, por tanto, nada.

NP	NBar	-	NP	
Det	Adj, NBar	Noun, Verb, NBar, VP	Det	Noun, NBar
the	old	man	a	ship

S → NP VP
NP → Det NBar
Nbar → Adj Noun
Nbar → man | men | ship | ships
Nbar → old
VP → man | mans
VP → Verb NP

Det → a | an | the
Adj → old
Noun → man | men | ship | ships
Verb → man | mans

Se rellena la última columna con negro para indicar que se ha finalizado, ya que no es posible generar una sucesión de 2 palabras con 'ship' como primera palabra, al ser el último término de la frase.

Paso 3

Se procede ahora a formar combinaciones de 3 palabras. Aquí es un poco más complejo: cuantas más palabras más combinaciones. Se puede combinar 1 símbolo que represente a la primera palabra con un símbolo que represente a las 2 palabras sucesivas, o bien el símbolo que represente las 2 primeras palabras con el símbolo de la tercera palabra.

- the old man =
  - 'the' + 'old man' = Det + NBar: salta la regla 'NP → Det NBar'
  - 'the old' + 'man' = NP + [Noun, Verb, NBar, VP]: salta la regla 'S→ NP VP'
- old man a =
  - 'old' + 'man a' = no se puede formar esta combinación, ya que 'man a' no tiene ningún valor (el valor '–' significa que no se encontró antes ningún símbolo).
  - 'old man' + 'a' = NBar + Det: no salta ninguna regla, ya que en ninguna parte derecha de las reglas se encontró 'NBar NP' ni 'NBar Det'.
- man a ship =
  - 'man' + 'a ship' = [Noun, Verb, NBar, VP] + NP: dispara la regla 'VP → Verb NP'
  - 'man a' + 'ship': de nuevo 'man a' no tiene ningún valor en la tabla

Se rellena con negro las 2 siguientes columnas:

NP, S	-	VP	[redacted]	[redacted]
NP	NBar	-	NP	[redacted]
Det	Adj, NBar	Noun, Verb, NBar, VP	Det	Noun, NBar
the	old	man	a	ship

S → NP VP
NP → Det NBar
Nbar → Adj Noun
Nbar → man | men | ship | ships
Nbar → old
VP → man | mans
VP → Verb NP

Det → a | an | the
Adj → old
Noun → man | men | ship | ships
Verb → man | mans

Paso 4

Combinaciones de 4 palabras.

- the old man a =
  - 'the' + 'old man a' = Det + '-' = ninguna regla disparada.
  - 'the old' + 'man a' = NP + '-' = ninguna regla disparada.
  - 'the old man' + 'a' = [NP,S] + [Det] = ninguna regla disparada.
- old man a ship =
- 'old' + 'man a ship' = [Adj, NBar] + VP = ninguna regla disparada.
- 'old man' + 'a ship' = NBar + NP = ninguna regla disparada.
- 'old man a' + 'ship' = '-' + [Noun, NBar] = ninguna regla disparada.

Se rellena con negro las 3 siguientes columnas:

-	-			
NP, S	-	VP		
NP	NBar	-	NP	
Det	Adj, NBar	Noun, Verb, NBar, VP	Det	Noun, NBar
the	old	man	a	ship

S → NP VP
NP → Det NBar
Nbar → Adj Noun
Nbar → man | men | ship | ships
Nbar → old
VP → man | mans
VP → Verb NP

Det → a | an | the
Adj → old
Noun → man | men | ship | ships
Verb → man | mans

Paso 5

Combinaciones de 5 palabras.

- the old man a ship =
  - 'the' + 'old man a ship' = Det + '-' = ninguna regla disparada.
  - 'the old' + 'man a ship' = NP + 'VP' = se dispara 'S → NP VP'. ¡Fin!

Llegados a este punto se ha finalizado, no se prueban, por tanto, las siguientes combinaciones porque ya se ha conseguido disparar una regla que logra asignar el símbolo inicial (S es el símbolo inicial) a todas las palabras de la frase (las 5 palabras). De no haber terminado, se debería continuar probando las combinaciones 'the old man'+'a ship' y 'the old man a'+'ship'.

Se rellena con negro las 4 siguientes columnas:

S	[redacted]	[redacted]	[redacted]	[redacted]
-	[redacted]	[redacted]	[redacted]	[redacted]
NP, S	-	VP	[redacted]	[redacted]
NP	NBar	-	NP	[redacted]
Det	Adj, NBar	Noun, Verb, NBar, VP	Det	Noun, NBar
the	old	man	a	ship

S → NP VP
NP → Det NBar
Nbar → Adj Noun
Nbar → man | men | ship | ships
Nbar → old
VP → man | mans
VP → Verb NP
Det → a | an | the
Adj → old
Noun → man | men | ship | ships
Verb → man | mans

Ahora solo hay que hacer el recorrido inverso al que se ha realizado para llegar hasta aquí, para sacar el árbol de análisis completo. Esto significa que, si se revisa en orden inverso todas las reglas que han ido saltando hasta llegar a la última, se pueden obtener todas las etiquetas morfosintácticas de nuestro análisis.

S → NP VP: S['*the old man a ship*']:
- **NP → Det NBar:** NP['*the old*']
  - · **Det → a | an | the:** Det['*the*']
  - · **NBar → old, Adj → old:** NBar[Adj['*old*']]
- **VP → Verb NP:** VP['*man a ship*']
  - · **Verb → man | mans:** Verb['*man*']
  - · **NP → Det NBar:** NP['*a ship*']
    - · **Det → a | an | the:** Det['a']
    - · **NBar → man | men | ship | ships , Noun → man | men | ship | ships:** NBar[Noun['ship']]

Obteniendo así el árbol completo:

```
S[
   NP[
     Det[the]
     Nbar[
            Adj[old]
         ]
     ]
   VP[
     Verb[man]
     Det[a]
     NBar[
            Noun[ship]
         ]
     ]
 ]
```

Hasta aquí se ha estudiado cómo convertir una CFG en CNF para después aplicar CKY. Pero existe otra forma alternativa del algoritmo CKY que no requiere pasar a CNF las reglas de tipo A → B. No se analizará, sin embargo, esta implementación, pues sobrepasa los límites de este curso introductorio, aunque es conveniente saber que existe.

9.2. Gramáticas de unificación y rasgos

Otro formalismo distinto de las CFG, que durante décadas ha sido ampliamente adoptado en lingüística computacional, son las gramáticas basadas en restricciones, también denominadas gramáticas de unificación y rasgos.

En lugar de utilizar únicamente símbolos atómicos, como en las CFG, usan parejas rasgo-valor para describir las unidades lingüísticas. Adicionalmente, se permite que dichas unidades estén anidadas: los valores de sus atributos pueden ser símbolos atómicos o una nueva estructura de rasgos.

A continuación, se muestra un ejemplo concreto de una estructura de rasgos que se llamará Z:

$$\begin{bmatrix} x0: \begin{bmatrix} cat & vp \\ head & [1] \end{bmatrix} \\ x1: \begin{bmatrix} cat & v \\ head & [1] \end{bmatrix} \end{bmatrix}$$

Figura 1.8. Estructura de rasgos. *Fuente: Handbook of NLP* (Chapman & Hall).

Se puede observar como esta estructura Z está formada a su vez por 2 estructuras de rasgos anidadas (x0 y x1). A su vez x0 contiene 2 parejas rasgo-valor: cat-vp y head-[1] y x1 contiene las parejas cat-v y head[1]. Así, los rasgos de una estructura pueden tomar valores atómicos (*cat* es un rasgo de *x0* que toma el valor atómico *vp*), pero también pueden contener otra estructura de rasgos (la estructura Z contiene 2 estructuras: x0 y x1).

A continuación, se verá otro ejemplo distinto, donde en cinco líneas se define lo que constituye una única regla gramatical, compuesta por una regla de producción inicial (X0 → X1 X2) más una serie de restricciones sobre los rasgos (concretamente 4 restricciones sobre los rasgos category y agreement).

$$
\begin{aligned}
&X_0 \rightarrow X_1\ X_2 \\
&\quad \langle X_0\ category \rangle = \mathrm{NP} \\
&\quad \langle X_1\ category \rangle = \mathrm{Det} \\
&\quad \langle X_2\ category \rangle = \mathrm{Noun} \\
&\quad \langle X_1\ agreement \rangle = \langle X_2\ agreement \rangle
\end{aligned}
$$

Figura 1.9. Gramática de restricciones. *Fuente: Handbook of NLP* (Chapman & Hall).

Lo que viene a significar esta regla es que un sintagma nominal (NP) está formado por un determinante (Det), seguido de un sustantivo (Noun), y que además el determinante y el adjetivo deberán concordar.

Otra forma alternativa, más sincrética e intuitiva de expresar la misma regla, sería, por ejemplo, así:

$$
\left[category : \mathrm{NP}\right] \rightarrow \left[\begin{matrix} category : & \mathrm{Det} \\ agreement : & \boxed{1} \end{matrix}\right] \left[\begin{matrix} category : & \mathrm{Noun} \\ agreement : & \boxed{1} \end{matrix}\right]
$$

Figura 1.10. Gramática de restricciones.
Fuente: Handbook of NLP (Chapman & Hall).

Se puede apreciar que, a diferencia de las CFGs, aquí, en las gramáticas de restricciones, los símbolos de la gramática ya no tienen por qué ser atómicos, sino que pueden ser rasgos (en la imagen anterior se aprecian 3 rasgos, cada uno de ellos limitado por sus respectivos corchetes).

Además, estas gramáticas permiten una operación básica que actúa sobre los rasgos: se trata de la operación de unificación. Consiste en fusionar aquellos términos compatibles en un nuevo término más genérico. De esta forma, 2 estructuras de rasgos (A y B) pueden unificarse en una nueva estructura (D = A *U* B) que combina la información de las 2 anteriores e incluye toda su información. A y B son más generales que D, o dicho de otra forma, A subsume a D y B subsume a D. No siempre es posible la unificación, como se aprecia en la figura 1.11., donde A *U* C no es posible al tener distinto rasgo de número (singular vs. plural):

The basic operation on feature terms is *unification*, which determines if two terms are compatible by merging them to the most general term compatible with both. As an example, the unification $A \sqcup B$ of the terms $A = [agreement : [number : \text{plural}]]$ and $B = [agreement : [gender : \text{neutr}]]$ succeeds with the result:

$$A \sqcup B = \left[agreement : \begin{bmatrix} gender : & \text{neutr} \\ number : & \text{plural} \end{bmatrix}\right]$$

However, neither A nor $A \sqcup B$ can be unified with

$$C = [agreement : [number : \text{singular}]]$$

Figura 1.11. Operación de unificación. *Fuente: Handbook of NLP* (Chapman & Hall).

De esta forma, las gramáticas de restricciones extienden y mejoran el formalismo de las CFG, siendo mucho más usadas en sintaxis computacional y haciendo innecesarias las CFG. Requieren codificar un menor número de reglas y computacionalmente son más eficientes.

Se acaban de ver dos formalismos distintos para implementar analizadores sintácticos (CFGs y gramáticas de unificación).

Pero, ¿cuáles son las características genéricas que todo buen analizador —en inglés se denomina parser— debería tener?

Robustez

Capacidad para gestionar entradas de texto no esperado. La robustez es una característica deseable en cualquier sistema NLP y para todas las fases de la cadena de procesamiento. Por ejemplo, un analizador morfológico deberá ser capaz de etiquetar palabras desconocidas y que no estén en su diccionario. Esta técnica se llama “word guessing” y es capaz, por ejemplo, de detectar que 'twerkear' es un verbo, por su terminación ('ar') y por el contexto de la frase. Esta misma robustez, deseable en los taggers léxicos, deberá tenerla el analizador sintáctico que se utilice. Entrarán al sistema que se utilice algunas frases cuya estructura no esté prevista en su gramática, bien porque el texto contenga errores o bien porque no se haya tenido en cuenta a la hora de construir la gramática. En esos casos, un sistema robusto deberá devolver siempre algo, no puede dejar de devolver un árbol sintáctico, igual que no se puede dejar de poner una etiqueta POS a la palabra 'twerkear'.

Poder de desambiguación

Es importante que un parser sea robusto ante cualquier texto inesperado. Pero, al mismo tiempo, no es deseable que devuelva cualquier análisis, sino el análisis más acertado entre todas las posibilidades que tiene que desambiguar. Una gramática que genere demasiadas posibilidades (overgeneration) será problemática y hará costosa la desambiguación, pero también es un problema el caso contrario (undergeneration). Para evitar estas situaciones, se suelen implementar gramáticas distintas y adaptadas para cada dominio específico (una gramática para textos científicos, otra para noticias de deportes, otra para conversaciones de adolescentes…). Otra posibilidad es ordenar todos los posibles análisis resultantes según un ranking probabilístico asociado a cada dominio: si es más probable encontrar un tipo de estructura sintáctica en un dominio específico y el texto de entrada pertenece a dicho dominio, la desambiguación deberá tener en cuenta dicha probabilidad.

Eficiencia

Un parser no solo debe ser robusto y preciso, además, debe ser eficiente. Según el algoritmo y el tipo de gramática que se escoja, el orden de magnitud del tiempo de ejecución variará. Por ejemplo, CKY es de orden cúbico O(n3), siendo n el número de palabras a analizar. Es decir, que si se tarda un segundo en analizar 10 palabras, se emplearán 8 segundos en analizar 20 palabras. Esto significa que en las frases muy largas, se empleará un tiempo muy superior al dedicado a las frases cortas —no es proporcional a la longitud —. Pero esto es la teoría, que tiene en cuenta siempre los peores casos. Normalmente, se suelen hacer mediciones prácticas con corpus reales para tener datos más reales del comportamiento de los parsers. Se ha demostrado, por ejemplo, que algunos parsers de unificación y rasgos, cuyo tiempo de ejecución teórico era de una magnitud exponencial O(an), en la práctica era cuadrático O(n2) con respecto a frases de un tamaño comprendido entre 1 y 30 palabras.

X. Análisis semántico

Esta última fase en la cadena de procesamiento NLP es con diferencia la menos estudiada y la más compleja de todas ellas. No existe un consenso académico en cuanto a la mejor forma de abordar este problema, que consiste básicamente en “entender” los textos de entrada al sistema.

Suelen usarse metalenguajes semánticos procesables computacionalmente. Es decir, que los sistemas traducen los textos de entrada a sentencias semánticas que adoptan unas convenciones determinadas, de forma parecida a los lenguajes de programación, que son lenguajes artificiales).

Anotación: Aplicaciones del análisis semántico

Las aplicaciones del análisis semántico son numerosas:

- Búsqueda y recuperación de información (en inglés, Information Retrieval: IR).
- Extracción de información (Information Extraction: IE).
- Resumen automático de textos.
- Data Mining.
- Traducción automática (Machine Translation: MT).
- Mejorar la compresión de “queries” de usuario.
- Mejorar las tecnologías asociadas a ontologías web.
- Mejorar los sistemas de representación del conocimiento (Knowledge Representation: KR).

Tradicionalmente, se hace una distinción entre semántica léxica, dedicada al estudio de la semántica atendiendo a la palabra individual, y semántica supraléxica, centrada en los aspectos gramaticales y de relaciones entre sintagmas y frases. Sin embargo, hoy en día hay un acuerdo generalizado en que ambas áreas del conocimiento están interrelacionadas.

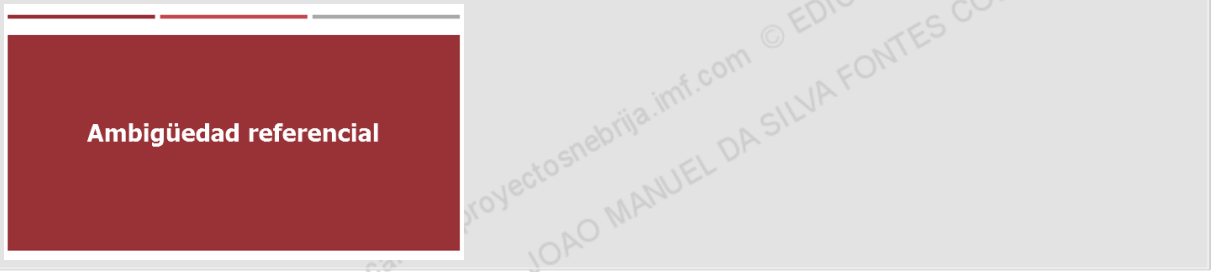
La semántica debe resolver la ambigüedad de significados y lo hace en tres niveles distintos:

Ambigüedad léxica

Una palabra puede tener varios significados (polisemia) y el sistema deberá escoger uno. También deberá resolverse el caso de palabras que se escriben o pronuncian igual, pero tienen distinto significado (homonimia): por ej. 'banco'=entidad financiera vs. 'banco'=asiento.

Ambigüedad de alcance

Algunas palabras como cuantificadores u operadores de negación (por ejemplo, 'no') pueden influir sobre distintos fragmentos de textos y el sistema deberá determinar sobre qué fragmento se aplica (ejemplo: 'yo no podría hacerlo' vs 'yo no ¿podría hacerlo otro?).



Hay palabras como los pronombres que apuntan a otras palabras y es necesario conocer a qué palabra aluden. Por ejemplo, en la frase 'John es bueno. Él me quiere', el pronombre 'él' hace referencia a hacia 'John'.

Existen diversas teorías relacionadas con la representación semántica, que básicamente pueden clasificarse en:

Formales vs. cognitivas

Mientras las teorías formales intentan construir precisos modelos matemáticos, usando, por ejemplo, la lógica formal, las teorías cognitivas surgieron más tarde como una reacción frente a las primeras y basándose en las ciencias cognitivas y la psicología.

Composicionales vs. léxicas

La semántica composicional trata de construir el significado de abajo a arriba: el significado de una frase es el significado de cada uno de sus componentes más las relaciones semánticas que se establecen entre dichos componentes. Pero no se entra en el significado en sí de las palabras, sino que se deja tal cual nos es dado —las aproximaciones de la lógica formal son un ejemplo de esta teoría—.

Por el contrario, las teorías léxicas se centran en el significado de cada uno de los componentes sin tener en cuenta la estructura en la que están inmersos. Existen dos subtipos: las teorías léxicas descomposicionales (analizan la estructura interna de cada componente, como se verá más adelante al estudiar los NSM) y las teorías léxicas relacionales (estudian las relaciones entre un componente léxico y el resto de componentes de un diccionario de significados, se verá al estudiar las ontologías y WordNet).

10.1. Lógica de predicados

La lógica de predicados es una aproximación composicional y formal al problema de construir un metalenguaje semántico que sirva de contenedor y nos permita modelizar los significados.

A continuación, se muestra un ejemplo concreto de esta vertiente semántica:

1. Some politicians are mortal.

∃x(politician(x) ^ mortal(x))

[There is an x (at least one)] so that x is a politician and x is mortal.]

2. All Australian students like Kevin Rudd.

∀x((student(x) ^ Australian(x)) → like(x,k))

[For all x with x being a student and Australian, x likes Kevin Rudd.]

Figura 1.12. Lógica de predicados.
***Fuente: Handbook of NLP*(Chapman & Hall).**

Esta representación usa los siguientes elementos:

- Variables: x
- Términos: un objeto o entidad concreta. En el ejemplo, k.
- Predicados: politician, mortal, like…
- Conectivas: ^ negación, → entonces
- Cuantificadores existenciales: ∃ algunos, ∀ todos

Además, la lógica de predicados incluye una serie de reglas de inferencia implícitas en estos sistemas que permiten inferir nuevo conocimiento a partir del introducido. Una regla muy conocida es el modus ponens:

a. Modus ponens:

(i) P (premise)

(ii) P → Q (premise)

(iii) Q (conclusion)

b. Ejemplo:

(i) Conrad is tired (P: tired(c))

(ii) Whenever Conrad is tired, he seeps (P:tired(c), Q:sleep(c), P->Q)

(iii) Conrad sleeps (Q: sleep(c))

Figura 1.13. Lógica de predicados: modus ponens.
***Fuente: Handbook of NLP*(Chapman & Hall).**

Así, si el sistema solo recibe como input 2 frases:

- "Conrad is tired" ("Conrad está cansado") y
- "Whenever Conrad is tired, he sleeps" ("Cuando Conrad está cansado, duerme"),

este podrá inferir automáticamente que Conrad duerme ("Conrad sleeps"). Inferir es parte del trabajo que como humanos hacemos al entender las frases y, por tanto, es muy adecuado que un sistema automático posea esta misma capacidad.

10.2. Teoría de representación del discurso

La teoría de representación del discurso (en inglés, Discourse Representation Theory: DRT) tiene en cuenta las frases en su contexto y no como elementos aislados. Nosotros, como humanos, somos capaces de interpretar una frase de distinta manera dependiendo del contexto del discurso —las frases precedentes—, esta teoría pretende modelizar esta capacidad.

A medida que oímos frases adicionales de un discurso, añadimos esa información dinámicamente a una denominada estructura de representación del discurso (Discourse Represantation Structure o DRS, en inglés).

Una DRS está formada por:

Referentes del discurso
Los objetos o entidades sobre los que se habla en un discurso.
Condiciones

Estas condiciones se aplican a los referentes y es la información que se va acumulando entorno a ellos a medida que se amplía el discurso. Se expresa mediante lógica de predicados. Utiliza, por tanto, la anterior teoría semántica que ya se ha visto, pero la amplía, dándole recursividad: unas DRS dentro de otras, como se puede ver en el siguiente gráfico, en que unas cajas incluyen a otras.

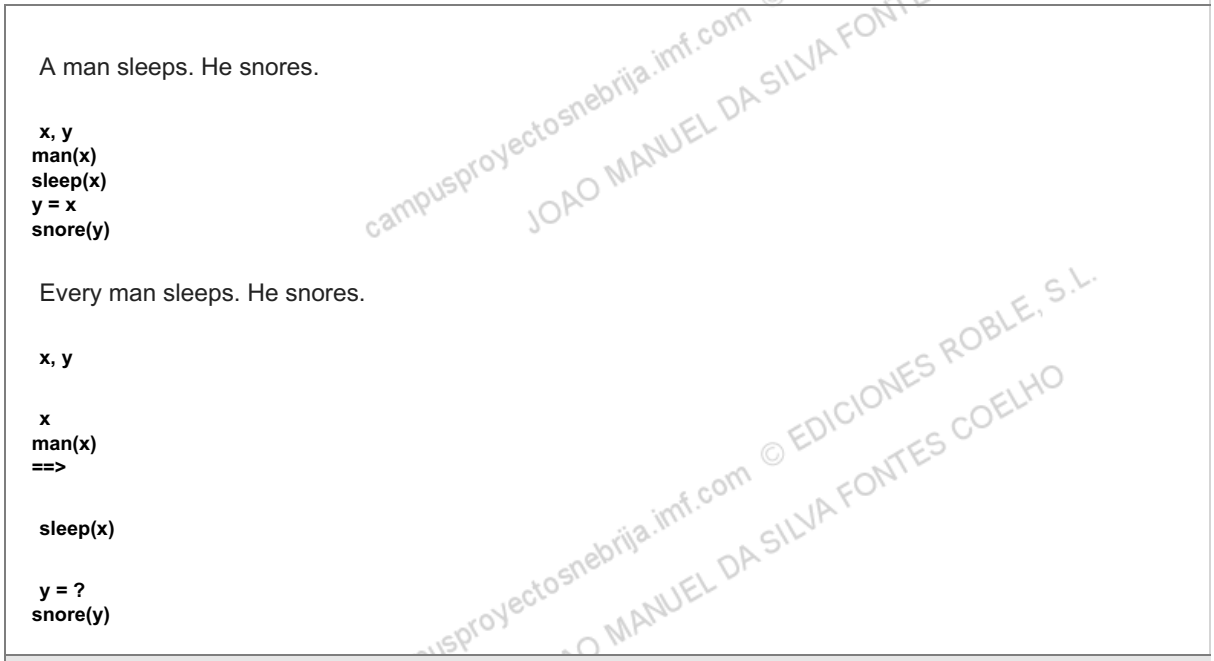


Figura 1.14. DRS. *Fuente: Handbook of NLP*(Chapman & H

Los discursos se representan gráficamente como cajas divididas en 2 secciones: la parte superior contiene los referentes y la inferior contiene las condiciones.

En el ejemplo anterior, el primer discurso es “A man sleeps. He snores”. Los 2 referentes del discurso son 'x' e 'y': 'a man' y 'He'. Las condiciones expresan la información del discurso:

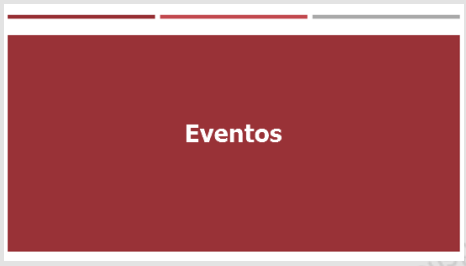
- man(x): 'x is a man'
- sleeps(x): 'x sleeps'
- y=x: y, es decir, 'He' apunta o se refiere a x, es decir, 'man'. Se ve aquí cómo se soluciona el problema semántico de la referencia entre pronombres y nombres (un pronombre apunta a un nombre).
- snores(y) 'y snores'.

El segundo discurso, “Every man sleeps. He snores”, es una muestra de cómo esta teoría consigue representar la recursividad, que es algo inherente al lenguaje natural.

10.3. Lexicón generativo

El lexicón generativo de Pustejovsky es otra teoría semántica dinámica. En lingüística, se suele entender por lexicón al listado de palabras que componen una lengua —una especie de diccionario—, por oposición a la gramática, que sería el conjunto de reglas de combinación de las palabras recogidas en el lexicón.

Pustejovsky postula que las palabras asumen distintos significados en distintos contextos. Por ejemplo, el adjetivo 'buen' en unos contextos puede significar algo positivo (“un buen marido”), mientras que en otros actúa a modo de intensificador (“un buen golpe”). Por ello, defiende que es necesario usar lexicones que recojan la representación semántica de todas las palabras en 4 niveles o estructuras:



Define el tipo de evento, sus estados, procesos, transiciones y subeventos.

Los distintos argumentos que aparecen en la estructura y sus funciones sintácticas.

Hay 4 tipos de qualia, constitutivo (de qué está hecho un objeto), formal (qué es ese objeto), télico (propósito o función del objeto) y agentivo (como se originó el objeto).

Relación de esta estructura léxica con otras estructuras del lexicón.

A continuación, se muestra un ejemplo concreto, la representación léxica del verbo 'build' (construir):

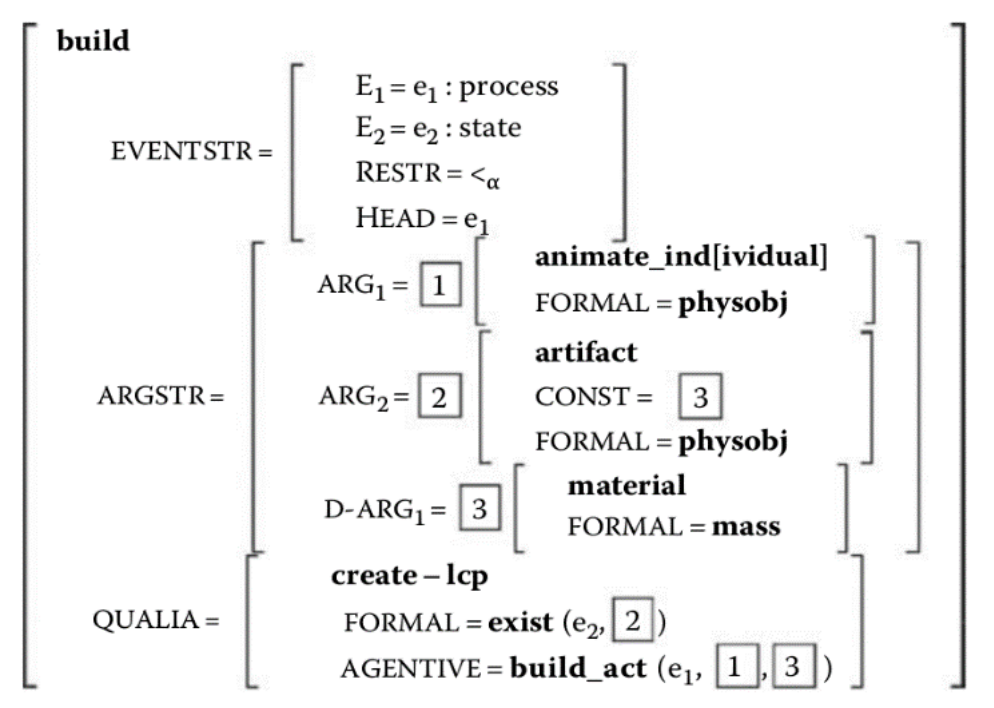


Figura 1.15. Representación léxica del verbo 'build'.
Fuente: Handbook of NLP (Chapman & Hall).

- **Eventos**: 'build' se define como un proceso e1 seguido de un estado resultante e2
- **Argumentos**: hay 3 argumentos. Si se atiende, por ejemplo, a la frase “John built a house out of bricks” (“Juan construyó una casa con ladrillos”), 'John' sería el ARG1 (un individuo animado), 'a house' sería el ARG2 (que está hecho de '3', es decir, de D-ARG1, o sea, de ladrillos) y 'out of bricks' sería el D-ARG1. D-ARG significa default argument, es decir, un argumento por defecto, que no siempre tiene por qué estar presente como realización sintáctica explícita, ya que no siempre se indica en inglés el material con el que se ha construido algo —en inglés se puede decir “John Built a house” sin necesidad de decir con qué material, pero no se puede decir “built a house” ni “John built”).
- **Qualia**: es un evento de tipo creativo (create-lcp), cuyo qualia agentivo indica que lo que se crea al construir surge de un acto de construcción (build-act), en el cual un proceso e1 realizado por 1 (la persona o individuo ARG1) usando 3 (el material D-ARG1). El qualia formal indica el resultado del acto de construcción, que es la existencia de un estado resultante e2 de la casa (2 = ARG2).

Se trata de una teoría descomposicional, pese a que al mismo tiempo posibilita la computación composicional del significado en contexto.

10.4. Metalenguaje semántico natural

Como su propio nombre indica, se trata de un metalenguaje, pero a diferencia de los anteriores metalenguajes que se han visto en otras teorías lingüísticas, la forma de expresarse de este se asemeja al lenguaje natural, en lugar de utilizar formulaciones lógicas, matemáticas o formales. Se trata de una teoría descompositiva y cognitiva.

Para expresar cada uno de los significados de los términos que define, utiliza unas estructuras denominadas explicaciones semánticas. Estas explicaciones hacen uso de unos universales semánticos (semantic primes) y de una sintaxis propia que combina estos universales. Los universales semánticos son 63 conceptos que han demostrado ser comunes a todos los idiomas y culturas:

TABLE 5.1 Semantic Primes, Grouped into Related Categories

I, YOU, SOMEONE, SOMETHING/THING, PEOPLE, BODY	Substantives
KIND, PART	Relational substantives
THIS, THE SAME, OTHER/ELSE	Determiners
ONE, TWO, SOME, ALL, MUCH/MANY	Quantifiers
GOOD, BAD	Evaluators
BIG, SMALL	Descriptors
KNOW, THINK, WANT, FEEL, SEE, HEAR	Mental predicates
SAY, WORDS, TRUE	Speech
DO, HAPPEN, MOVE, TOUCH	Actions, events, movement, contact
BE (SOMEWHERE), THERE IS, HAVE, BE (SOMEONE/SOMETHING)	Location, existence, possession, specification
LIVE, DIE	Life and death
WHEN/TIME, NOW, BEFORE, AFTER, A LONG TIME, A SHORT TIME, FOR SOME TIME, MOMENT	Time
WHERE/PLACE, HERE, ABOVE, BELOW, FAR, NEAR, SIDE, INSIDE	Space
NOT, MAYBE, CAN, BECAUSE, IF	Logical concepts
VERY, MORE	Intensifier, augmentor
LIKE/WAY	Similarity

Notes: Primes exist as the meanings of lexical units (not at the level of lexemes). Exponents of primes may be words, bound morphemes, or phrasemes. They can be formally complex. They can have combinatorial variants (allolexes). Each prime has well-specified syntactic (combinatorial) properties.

Figura 1.16. Primitivas semánticas.
Fuente: Handbook of NLP (Chapman & Hall).

Pero estos 63 conceptos son insuficientes para expresar algunos significados y, por ello, se han definido unas 250 moléculas semánticas (semantic molecules), que facilitan la elaboración de explicaciones semánticas. Por ejemplo, la molécula 'pájaro' permite definir significados como 'águila' o 'loro'.

A continuación, se muestra un ejemplo concreto de dos explicaciones semánticas: una para la palabra 'sad' (triste) y otra para la palabra 'unhappy' (descontento).

[B] Semantic explication for Someone X felt sad

a.

someone X felt something bad

b.

like someone can feel when they think like this:

“I know that something bad happened

I don’t want things like this to happen

I can’t think like this: I will do something because of it now

I know that I can’t do anything”

[C] Semantic explication for Someone X felt unhappy

a.

someone X felt something bad

b.

like someone can feel when they think like this:

“some bad things happened to me

I wanted things like this not to happen to me

I can’t not think about it”

c.

this someone felt something like this, because this someone thought like this

Figura 1.17. Explicaciones semánticas de 'sad' y 'unhappy'.
***Fuente: Handbook of NLP*(Chapman & Hall).**

Como se puede apreciar, ambas son muy similares, pero 'unhappy' tiene un componente adicional (c.) que hace hincapié en el aspecto cognitivo de estar descontento: es posible que estemos tristes sin saber por qué, pero cuando estamos descontentos es por algún motivo que conocemos y acerca del cual tenemos pensamientos. Por tanto, este metalenguaje permite expresar aspectos cognitivos muy sutiles.

A modo de curiosidad, obsérvense las diferencias entre culturas e idiomas, si se comparan los dos términos anteriores con sus equivalentes en chino. Se puede ver que son similares, pero no significan exactamente lo mismo: 'bei' es más intenso y fatalista y está relacionado con la inevitabilidad de un destino, mientras que 'chou' no tiene esa carga existencial, es más bien una mezcla entre 'sad' y 'worried.' Es decir, ninguno de los dos equivale plenamente a 'sad'.

> [D] Semantic explication for Someone X felt bei [Chinese]
>
> a.
>
> someone X felt something very bad
>
> b.
>
> like someone can feel when they think like this:
>
> “something bad happened now
>
> I know that after this good things will not happen anymore
>
> I don’t want things like this to happen
>
> I want to do something if I can
>
> I know that I can’t do anything
>
> because I know that no one can do anything when things like this happen”
>
> c.
>
> this someone felt something like this, because this someone thought like this
>
> [E] Semantic explication for Someone X felt chou [Chinese]
>
> a.
>
> someone X felt something bad
>
> b.
>
> like someone can feel when they think like this:
>
> “something bad is happening to me
>
> before this, I did not think that this would happen to me
>
> I don’t want things like this to happen to me
>
> because of this, I want to do something if I can
>
> I don’t know what I can do

I can’t not think about this all the time”

c.

this someone felt something like this, because this someone thought like this

Figura 1.18. Explicaciones semánticas de 'bei' y 'chou'. *Fuente: Handbook of NLP* (Chapman & Hall).

10.5. Semántica orientada a objetos

La orientación a objetos es una técnica proveniente de la Inteligencia Artificial, que hoy en día está muy extendida en el mundo de la programación.

Lenguajes como Python o Java permiten definir estructuras de datos y métodos (funciones) que tratan de modelizar el mundo real, haciendo los programas más fáciles de entender y mantener. Imita la forma en que los humanos gestionamos y percibimos la información: los objetos son entidades del mundo real (cosas, personas…) con unos atributos específicos y sobre ellos se efectúan una serie de operaciones específicas. Así, por ejemplo, si se modeliza una entidad financiera, existirá un objeto 'cliente' que tendrá los atributos 'nombre', 'apellido', 'edad', 'dni' y tendrá un método específico 'ListarMovimientos()' para todos los clientes.

Suele usarse para modelizar verbos y su representación gráfica (denominada UER: Unified Eventity Representation) está basada en el conocido estándar UML (Unified Modeling Language). A continuación, se muestra un ejemplo de UER donde se representa el verbo 'wake up' (despertarse). Una persona X despierta a otra persona Y, lo cual genera un estado de despierto (awake) en Y:

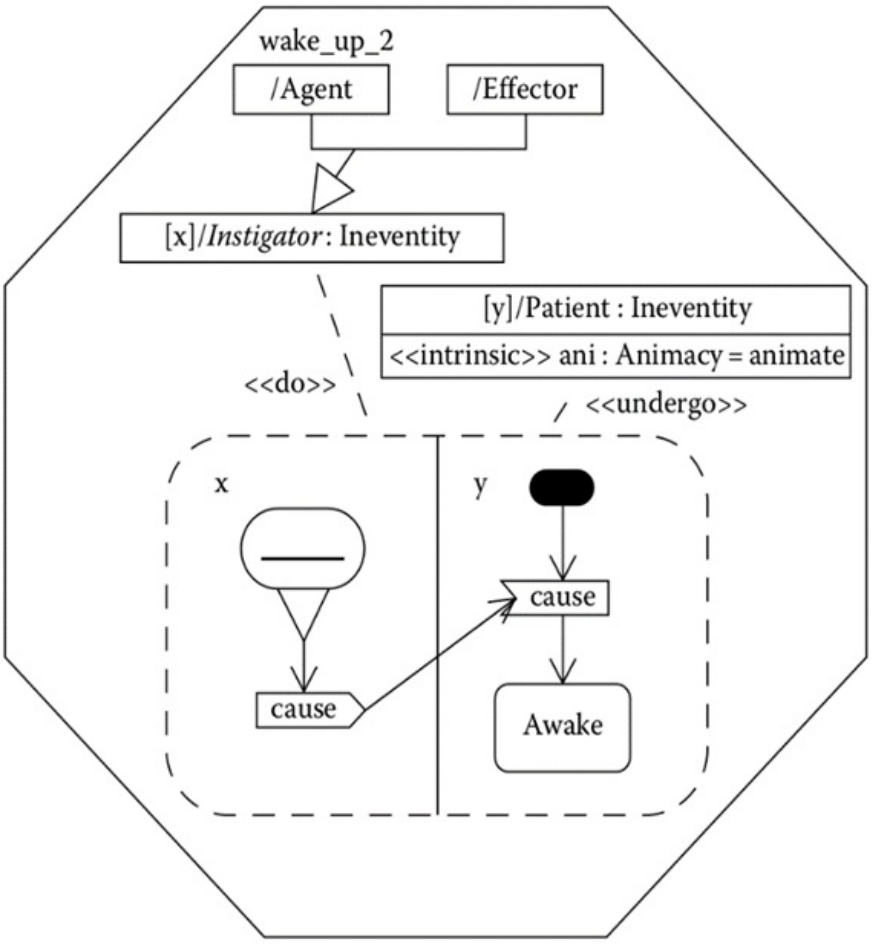


Figura 1.19. Diagrama UER del verbo 'wake up'. *Fuente: Handbook of NLP* (Chapman & Hall).

10.6. Relaciones semánticas y ontologías

Relaciones semánticas

Las relaciones semánticas son relaciones entre elementos léxicos que forman redes de palabras relacionadas por sus significados. Un buen ejemplo de ellos es Wordnet, una base de datos léxica ampliamente usada en sistemas NLP y en tareas como la extracción de información y recuperación de información.

Se suelen distinguir dos tipos de relaciones semánticas:

Relaciones horizontales

- **Sinonimia**: palabras con el mismo significado.La sinonimia puede ser total —son pocas las palabras con sinonimia total— o parcial —'viejo' y 'anciano' son sinónimos parciales, ya que 'anciano' tiene una connotación adicional de respeto—.
- **Oposición o antonimia:** palabras con significados opuestos. Se distinguen los siguientes tipos:
- **Antónimos graduales**: oposición gradual, representa grados entre los extremos opuestos de una escala. Así, entre las palabras '*frío*' y '*caliente*' se puede establecer la gradación frío-fresco-tibio-cálido-caliente.
- **Antónimos complementarios**: la oposición entre el significado de dos palabras no admite gradación, son totalmente incompatibles: *'sano|enfermo'*, *'encendido|apagado'. La incompatibilidad implica palabras del mismo campo semántico pero cuyo significado excluye el significado de las otras: 'blanco/negro'. No necesariamente tienen que ser antónimos, por ejemplo: 'perro/caballo' pertenecen al campo semántico 'animales'.*
- **Antónimos recíprocos o relacionales** (en inglés, conversity):al sustituir uno por otro es obligatorio cambiar el orden sintáctico en que aparecen en la frase las cosas o personas relacionadas: *'suegro|yerno'*, *'comprar|vender'*, *'mayor que|menor que'*, *'delante de|detrás de'*.

Relaciones verticales

- **Hiponimia:** se produce cuando un concepto pertenece a una categoría más genérica. Por ejemplo: perro->animal.
- **Meronimia:** un concepto es parte de otro. Ejemplo: mano->cuerpo.
- **Troponimia:** parecido a la hiponimia, pero para verbos: un verbo lleva a cabo la acción de otro verbo. Ejemplo: balbucear->hablar.

Relaciones ontológicas

Son muy similares a las relaciones semánticas, pero no son exactamente lo mismo, ya que son relaciones entre significados, independientemente de que esos significados tengan o no una palabra asociada. Las ontologías son, por tanto, redes de conceptos relacionados, mientras que las redes semánticas son redes de palabras relacionadas.

La semántica ontológica es una teoría del significado que hace uso de las ontologías para representar los textos. Modeliza el conocimiento del mundo mediante ontologías, de forma que el conocimiento que extrae de los textos es usado para razonar e inferir nuevo conocimiento. También se usa esta teoría para representar el conocimiento en sistemas de generación de lenguaje natural.

10.7. Roles semánticos

Las ontologías y las redes semánticas que se acaban de ver aportan una visión muy estática y centrada en palabras y conceptos aislados. Sin embargo, el lenguaje natural es mucho más abierto y expresivo y requiere modelizar complejas situaciones y descripciones. En la frase “el perro come patatas”, se pueden extraer 3 palabras y sus relaciones semánticas (perro->animal, come->alimentar, patatas->vegetal). Pero si se desea profundizar más en la comprensión de las estructuras y relaciones semánticas subyacentes, hay que usar otro tipo de relaciones.

Los roles semánticos son las relaciones entre predicados y argumentos de las frases y están muy centradas en el verbo. Volviendo al ejemplo anterior, es posible apreciar que en torno al verbo 'comer' está el rol del comensal (el que come) y el rol de la comida (aquello que come).

Se ha destacado WordNet en el apartado anterior como ejemplo de red de relaciones semánticas y el proyecto FrameNet sería su equivalente en el campo de los roles semánticos. En FrameNet, los verbos serían los denominados Lexical Units, mientras que los roles serían los Frame Elements. De esta forma, extraer los roles semánticos en la frase anterior sería algo parecido a esto:

Frame: ingerir

LU: comer

FE comensal: el perro

FE comida: patatas

Algunos teóricos han definido un grupo muy reducido de roles semánticos, como se muestra en la figura 1.20., pero estas teorías se alejan de la visión que tiene FrameNet, con un número muchísimo más amplio de roles.

TABLE 5.2 Semantic Roles and Their Conventional Definitions

Role	Description
agent	a wilful, purposeful instigator of an action or event
effector	the doer of an action, which may or may not be wilful or purposeful
experiencer	a sentient being that experiences internal states, such as perceivers, cognizers, and emoters
instrument	a normally inanimate entity manipulated by an agent in carrying out an action
force	somewhat like an instrument, but it cannot be manipulated
patient	a thing that is in a state or condition, or undergoes a change of state or condition
theme	a thing which is located or is undergoing a change of location (motion)
benefactive	the participant for whose benefit some action is performed
recipient	someone who gets something (recipients are always animate or some kind of quasi-animate entity)
goal	destination, which is similar to recipient, except that it is often inanimate
source	the point of origin of a state of affairs
location	a place or a spatial locus of a state of affairs
path	a route

Figura 1.20. Roles semánticos.
Fuente: Handbook of NLP (Chapman & Hall).

XI. Resumen

El PLN se basa en la creación de cadenas de procesamiento, que consisten en la sucesión de algoritmos a un texto inicial de entrada:

- División de frases: se parte un texto en frases.
- Tokenización: se divide una frase en tokens.
- Análisis léxico.
- Análisis morfológico: se asigna una etiqueta morfológica a los tokens.
- Lematización: se asigna el lema a cada token.
- Análisis sintáctico: creación de árboles sintácticos.
- Análisis semántico: se añade significado a los árboles sintácticos y se desambigua entre distintos significados.
- Análisis pragmático: tiene en cuenta el contexto más amplio de la frase para añadir significado.

XII. Bibliografía

La mayor parte del material de este documento proviene del libro de referencia de la asignatura: Indurkhya, N. y Damerau, F. J. *Handbook of Natural Language Processing*. Chapman & Hall, 2ª ed.; 2010. Algunos fragmentos de este libro se pueden encontrar en Google Books.

El alumno, para poder responder correctamente a todas las preguntas del test teórico, deberá visitar dos webs relacionadas con la semántica:

- FrameNet
- WordNet