Maestría en Inteligencia Artificial

Procesamiento del Lenguaje Natural

Tema 1. Introducción al procesamiento del lenguaje natural

Índice

Esquema

PROCESAMIENTO DEL LENGUAJE NATURAL

Campo de la inteligencia artificial que tiene como objetivo que las máquinas sean capaces de realizar tareas que involucren el lenguaje humano.

Historia del procesamiento del lenguaje natural

- Paradigmas fundacionales: década de 1940 y 1950
- Paradigmas simbólico y estocástico: 1957-1970
- Cuatro paradigmas de investigación: 1970-1983
- Revivir del empirismo y de los modelos de estados finitos: 1983-1993
- Unión de las diferentes vertientes: 1994-1999
- Auge del aprendizaje automático: 2000

Aplicaciones del procesamiento del lenguaje natural

- Agentes conversacionales
- Traducción automática
- Búsqueda de respuestas *(Question Answering)*

Conocimiento del lenguaje utilizado en el procesamiento del lenguaje natural

- **Fonética y fonología**: conocimiento sobre los sonidos lingüísticos.
- **Morfología**: conocimiento de los componentes significativos de las palabras.
- **Sintaxis**: conocimiento de las relaciones estructurales entre palabras.
- **Semántica**: conocimiento del significado.
- **Pragmática**: conocimiento de la relación del significado con los objetivos y las intenciones.
- **Discurso**: conocimiento sobre unidades lingüísticas más grandes que un solo enunciado.

Ideas clave

1.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Este tema comienza introduciendo de manera general el campo de la inteligencia artificial denominado procesamiento del lenguaje natural (PLN). La idea de que las máquinas sean capaces de poseer habilidades de comunicación y lenguaje es tan antigua como la propia aparición de los primeros ordenadores.

En los siguientes apartados se describe la historia del procesamiento del lenguaje natural y se puede observar cómo ha evolucionado dicho campo y los diferentes enfoques que ha ido tomando a lo largo de la historia. Además, en este tema también se describen diferentes aplicaciones que utilizan técnicas basadas en el procesado del lenguaje natural, por ejemplo, los agentes conversacionales o sistemas de diálogo que se van a estudiar en profundidad en capítulos posteriores de esta asignatura, la traducción automática, la búsqueda de respuestas o la corrección ortográfica y la verificación gramatical. Por último, analizamos cómo el conocimiento del lenguaje es imprescindible para el correcto funcionamiento de los sistemas de procesamiento de lenguaje natural, describiendo los diferentes tipos de conocimiento del lenguaje.

Los apartados de los que consta este tema son:

- Procesamiento del lenguaje natural (PLN).
- Historia del procesamiento del lenguaje natural.
- Aplicaciones.
- Conocimiento del lenguaje utilizado en el PLN.

Al finalizar el estudio de este tema serás capaz de:

- Definir el concepto de procesamiento del lenguaje natural.
- Narrar la historia del PLN.
- Identificar posibles aplicaciones en las que se utiliza PLN.
- Definir los diferentes tipos de conocimiento del lenguaje que es necesario para correcto funcionamiento de los sistemas de PLN.

Accede a los ejercicios de autoevaluación a través del aula virtual

1.2. Procesamiento del lenguaje natural

Accede al vídeo «Historia del procesamiento del lenguaje natural» a través del aula virtual

El procesamiento del lenguaje y del habla ha sido tratado históricamente de forma muy diferente en la informática, la ingeniería, la lingüística, la psicología o la ciencia cognitiva. Hoy en día se concibe el procesamiento del lenguaje natural como área que abarca varios campos diferentes y diversos pero superpuestos. Por ello, el procesamiento del lenguaje natural es un campo interdisciplinario que une a informáticos, ingenieros electrónicos y de telecomunicaciones con lingüistas, sociólogos y psicólogos.

El reconocimiento de la voz, que incluye tareas del procesamiento de la señal, se ha tratado tradicionalmente en la ingeniería electrónica y de telecomunicaciones. El análisis sintáctico y la interpretación semántica de las palabras y frases son áreas tradicionales del procesamiento del lenguaje natural que se estudian en el campo de la informática. La morfología, la fonología y la pragmática son tareas de investigación en la lingüística computacional. Psicolingüistas y sociolingüistas estudian respectivamente los mecanismos cognitivos para la adquisición del lenguaje y cómo la sociedad influye en el uso de la lengua.

El ancho espectro que abarca el campo del procesamiento del lenguaje natural hace que se conozca con diferentes nombres debido a las diferentes vertientes involucradas. Algunos de estos nombres, que provienen de estas diferentes facetas, serían procesamiento del lenguaje y del habla, tecnología del lenguaje, procesamiento del lenguaje natural, lingüística computacional o reconocimiento y síntesis del habla.

En la inteligencia artificial, el procesamiento del lenguaje natural es un campo que tiene como objetivo que las máquinas sean capaces de realizar tareas que involucren el lenguaje humano.

Algunas de las tareas que debe realizar una máquina para ser capaz de procesar el lenguaje natural incluyen funcionalidades tales como la de habilitar a la máquina de habilidades para comunicarse con personas, la de mejorar la comunicación entre humanos o, simplemente, la de procesar un texto o el habla.

Accede a los ejercicios de autoevaluación a través del aula virtual

1.3. Historia del procesamiento del lenguaje natural

Accede al vídeo «Historia del procesamiento del lenguaje natural» a través del aula virtual

La historia del procesamiento del lenguaje natural se puede dividir en diferentes etapas. Desde la aparición de las bases o paradigmas fundacionales en la década de 1940 hasta la explotación de los paradigmas más modernos para desarrollar hoy en día aplicaciones más inteligentes.

1940-1950	Paradigmas fundacionales
1957-1970	Paradigma simbólico y estocástico
1970-1983	Cuatro paradigmas de investigación
1983-1993	Revivir del empirismo y los modelos de estados finitos
1994-1999	Unión de las diferentes vertientes
2000	Auge del aprendizaje automático

Tabla 1. Cronología de las diferentes etapas de la historia del PLN.

Paradigmas fundacionales: década de 1940 y 1950

El principio del PLN data del período justo después de la II Guerra Mundial, cuando se dio el origen del ordenador. En este período, desde la década de 1940 hasta el final de la década de 1950, se trabajó intensamente en dos paradigmas fundacionales:

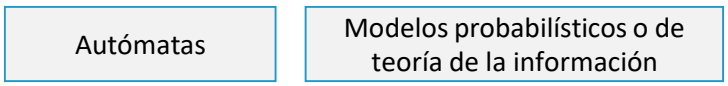


Figura 1. Paradigmas fundacionales.

Autómatas

El autómata surgió en la década de 1950 a partir del famosísimo estudio publicado por Turing (1936), *Los números computables, con una aplicación al Entscheidungsproblem*, y que se considera como la base de la informática moderna. El trabajo de Turing condujo primero a un modelo simplificado de la neurona como elemento de computación que podría describirse en términos de lógica proposicional (McCulloch y Pitts, 1943) y luego a la definición de los **autómatas finitos** y las **expresiones regulares** (Kleene, 1951).

Shannon aplicó las cadenas de Markov, un modelo de proceso estocástico discreto en el que la probabilidad de ocurrencia de un evento depende solo del evento

inmediatamente anterior, a los autómatas para el lenguaje (Shannon, 1948). Aprovechando las ideas del trabajo de Shannon, Chomsky fue el primero en considerar las máquinas de estados finitos como una forma de caracterizar una gramática y definió el lenguaje de estados finitos como un lenguaje generado por una gramática de estados finitos (Chomsky, 1956).

Estos primeros modelos llevaron a la aparición de la **teoría del lenguaje formal**, que utilizó el álgebra y la teoría de conjuntos para definir los lenguajes formales como secuencias de símbolos. Esta teoría incluye las gramáticas libres de contexto, un concepto que Chomsky definió en 1956 para las lenguas naturales, pero que también fue descubierto de forma independiente por Backus y Naur en su descripción del lenguaje de programación ALGOL (Backus, 1959) (Naur et al., 1960).

Teoría de la información

El segundo paradigma fundamental de este período fue el desarrollo de algoritmos probabilísticos para el procesamiento del lenguaje y del habla. Esta idea proviene de la otra gran contribución de Shannon, el teorema de codificación de canal en la **teoría de la información,** y que muestra que es posible la transmisión y decodificación del lenguaje a través de un canal de comunicación ruidoso.

Shannon tomó prestado el concepto de **entropía** de la termodinámica como una forma de medir la capacidad de información de un canal o el contenido de información de un idioma, y realizó la primera medida de la entropía del inglés utilizando técnicas probabilísticas.

También durante este período inicial se desarrolló el espectrógrafo de sonido y se realizó investigación fundamental en la fonética instrumental, lo que sentó las bases para el **reconocimiento de la voz**. La primera máquina capaz de realizar el reconocimiento de la voz apareció a principios de los años cincuenta. En 1952 investigadores de Bell Labs construyeron un sistema estadístico basado en correlación que podía reconocer con un 97-99 % de precisión cualquiera de los diez

primeros números a partir de unos patrones grabados con los sonidos de las vocales de un único hablante (Davis, Biddulph y Balashek, 1952).

Paradigma simbólico y estocástico: 1957-1970

A fines de la década de 1950 y comienzos de la década de 1960, el procesamiento del habla y el lenguaje se había dividido muy claramente en dos paradigmas:

Figura 2. Paradigmas a partir de la década de los 60.

El paradigma simbólico

Apareció de dos líneas de investigación: la teoría del lenguaje formal y la inteligencia artificial.

La primera línea se basaba en el trabajo que realizaron Chomsky y sus colaboradores en la **teoría del lenguaje formal y la sintaxis generativa**, además de en el trabajo de muchos lingüistas e informáticos que estudiaban los algoritmos de análisis, inicialmente de arriba hacia abajo (*top-down*) y de abajo hacia arriba (*bottom-up*) y luego a través de la programación dinámica. Uno de los primeros sistemas de análisis fue TDAP (*Transformations and Discourse Analysis Project*) que implementó Zelig Harris entre junio de 1958 y julio de 1959 en la Universidad de Pennsylvania.

La segunda línea de investigación del paradigma simbólico fue el nuevo campo de la **inteligencia artificial**. En el verano de 1956 John McCarthy, Marvin Minsky, Claude Shannon y Nathaniel Rochester congregaron durante dos meses a un grupo de investigadores para realizar un simposio sobre lo que decidieron llamar «inteligencia artificial». Aunque el incipiente ámbito de la inteligencia artificial incluía una minoría de investigadores centrados en algoritmos estocásticos y estadísticos (incluidos los modelos probabilísticos y las redes neuronales), este nuevo campo se enfocó

básicamente en el razonamiento y la lógica, representados por el trabajo de Newell y Simon en los programas de ordenador Logic Theorist y General Problem Solver (Newell, Shaw y Simon, 1959).

En los principios de la inteligencia artificial fue cuando se construyeron los primeros **sistemas de comprensión del lenguaje natural**. Estos sistemas simples estaban diseñados para trabajar en UN dominio concreto y basaban su funcionamiento en la búsqueda de patrones y heurística de palabras clave para realizar el razonamiento y la búsqueda de respuestas. Fue a finales de la década de 1960 cuando se desarrollaron algunos sistemas lógicos más formales.

El paradigma estocástico

Se desarrolló principalmente por parte de investigadores de los departamentos de estadística y de ingeniería electrónica. A fines de la década de 1950 comenzó a aplicarse el método bayesiano al problema del reconocimiento óptico de caracteres. Por ejemplo, Bledsoe y Browning construyeron un **sistema bayesiano** de reconocimiento de texto que calculaba la probabilidad de una secuencia de letras dadas las palabras de un diccionario.

En la década de 1960 aparecieron también los **primeros modelos psicológicos** para el PLN basados en gramáticas transformacionales y los primeros corpus disponibles *online*. Un ejemplo es el Brown Corpus, un corpus del inglés americano desarrollado en 1963 por la Brown University y que contenía una colección de un millón de palabras extraídas de 500 textos de diferentes géneros: periódicos, novelas, no ficción, académico, etc. (Kucera y Francis, 1967).

Cuatro paradigmas de investigación: 1970-1983

En el siguiente período, en la década de 1970 y a principios de la década de 1980, se produce una explosión de la investigación en el procesamiento de lenguaje y del

habla. Es en esta época cuando se desarrollan una serie de paradigmas de investigación que todavía hoy dominan el campo:

Figura 3. Paradigmas de investigación dominantes.

El **paradigma estocástico** jugó un papel muy importante en el desarrollo de algoritmos de reconocimiento de voz para los que se utilizaban modelos ocultos de Markov (HMM) y el teorema de codificación de canal de Shannon. Algunos de los investigadores que trabajaron en este ámbito fueron Jelinek, Bahl, Mercer y sus socios del Thomas J. Watson Research Center de IBM, Baker en la Carnegie Mellon University, e investigadores de los Bell Laboratories de AT&T.

El **paradigma basado en lógica** surgió del trabajo de Alain Colmerauer y sus colaboradores en la década de 1970 que desarrollaron Q-system, un analizador *bottom-up* basado en una serie de reglas con variables lógicas y que permitían la traducción del inglés al francés (Colmerauer, 1978). En este paradigma basado en lógica se engloba también la gramática de cláusulas definidas (*Definite Clause Grammar*, DCG), una forma de expresar la gramática en un lenguaje de programación lógico como por ejemplo Prolog (Pereira y Warren, 1980).

De forma independiente, apareció también el trabajo de Kay (1979) sobre la gramática funcional y, poco después, el trabajo de Joan Bresnan y Ronald Kaplan (1982) sobre la gramática léxico funcional (*Lexical functional grammar*, LFG), una gramática generativa que se centra en la investigación de la sintaxis del lenguaje natural.

El **paradigma de la comprensión del lenguaje natural** apareció durante la década de 1970 con la aparición del sistema SHRDLU (Winograd, 1972). Este sistema simulaba un robot que movía bloques y era capaz de recibir comandos del lenguaje natural en

formato de texto, como por ejemplo «mueve el bloque rojo que se encuentra en la parte superior del verde más pequeño». Este sistema, complejo y sofisticado para su época, también fue el primero en construir una gramática relativamente extensa del inglés.

Los avances en el modelo de análisis del lenguaje de Winograd dejó claro que la investigación debía comenzar a enfocarse en la semántica y los modelos de discurso. Roger Schank y sus colaboradores, conocidos como la Escuela de Yale, construyeron una serie de programas de comprensión del lenguaje que se centraron en el conocimiento humano, por ejemplo, en planes y objetivos, y la organización de la memoria humana (Schank y Riesbeck, 1981). Estos trabajos, por ejemplo el realizado por R. F. Simmons (1973), usaban una representación del conocimiento en forma de red, lo que se conoce como redes semánticas (Quillian, 1968), y comenzaron a incorporar en sus representaciones la idea de gramática de casos (Fillmore, 1968), por la cual se establece una relación entre un verbo y múltiples papeles temáticos que serían los sintagmas nominales.

El paradigma basado en lógica y el paradigma de la comprensión del lenguaje natural se unificaron en sistemas que usaban la lógica de predicados como una representación semántica, como el sistema de búsqueda de respuestas LUNAR (Woods, 1967).

El **paradigma del modelado del discurso** se centró en las cuatro áreas clave del discurso. Grosz y sus colaboradores introdujeron el estudio de la estructura del discurso y el enfoque del discurso (Grosz, 1977) (Grosz y Sidner, 1986). Una serie de investigadores comenzaron a trabajar en la resolución de forma automática de referencias en el discurso (Hobbs, 1978) (Hobbs, 1979). Por último, se desarrolló el modelo BDI (creencias-deseos-intenciones en inglés), un marco de trabajo basado en la lógica de actos de habla (Perrault y Allen, 1980) (Cohen y Perrault, 1979).

Revivir del empirismo y los modelos de estados finitos: 1983-1993

A partir de 1983 volvieron dos clases de modelos que habían perdido popularidad a finales de la década de 1950 y principios de la década de 1960 debido a los argumentos teóricos en su contra (Chomsky, 1959).

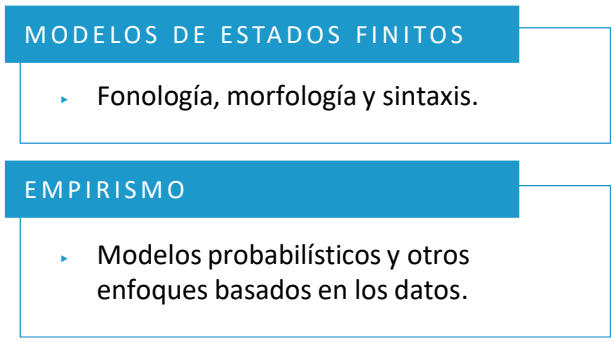


Figura 4. Modelos de estados finitos y modelos empíricos.

Los **modelos de estados finitos** revivieron en esta época y comenzaron a recibir atención porque se usaron en la fonología y la morfología (Kaplan y Kay, 1981), y en la sintaxis (Church, 1980).

La segunda tendencia en este período fue lo que se ha llamado el **retorno del empirismo**, marcada por el aumento de los modelos probabilísticos para el procesamiento del lenguaje y del habla. Cabe destacar la influencia del trabajo de los investigadores del Thomas J. Watson Research Center de IBM sobre modelos probabilísticos en el reconocimiento de voz. Estos métodos probabilísticos y otros enfoques basados en los datos se extendieron al etiquetado morfosintáctico (POS *tagging*), al análisis y resolución de ambigüedades y a la semántica.

Este paradigma empírico vino acompañado por el enfoque de la evaluación de los modelos basado en los datos. Entonces en este período se desarrollaron métricas cuantitativas para la evaluación y se enfatizó en la comparación del rendimiento de estas métricas con los resultados de las investigaciones previas. Además, en este período, se trabajó considerablemente en la generación de lenguaje natural.

Unión de las diferentes vertientes: 1994-1999

En los últimos cinco años del pasado milenio el campo del procesamiento del lenguaje natural sufrió grandes cambios.

En primer lugar, los modelos probabilísticos y los modelos basados en datos se volvieron estándares para el procesamiento del lenguaje natural. Los algoritmos de análisis, de etiquetado morfosintáctico, de resolución de referencias y de procesamiento del discurso empezaron a incorporar probabilidades y adoptaron metodologías de evaluación provenientes de los ámbitos del **reconocimiento de la voz** y **la recuperación de información**.

En segundo lugar, el aumento en la velocidad y la memoria de los ordenadores permitió la explotación comercial de varias áreas del procesamiento del lenguaje y del habla, en particular el reconocimiento de la voz y la revisión de la ortografía y la gramática. Además, los algoritmos de procesamiento del lenguaje y del habla comenzaron a aplicarse a la comunicación aumentativa para ayudar a personas con algún tipo de discapacidad. Por último, el aumento de la web enfatizó la necesidad de la recuperación y la extracción de información basada en el lenguaje natural.

Auge del aprendizaje automático: 2000

Las tendencias empiristas que marcaron la última parte de la década de 1990 se aceleraron a un ritmo asombroso en el nuevo milenio. Esta aceleración fue impulsada en gran parte por el auge del **aprendizaje automático**.

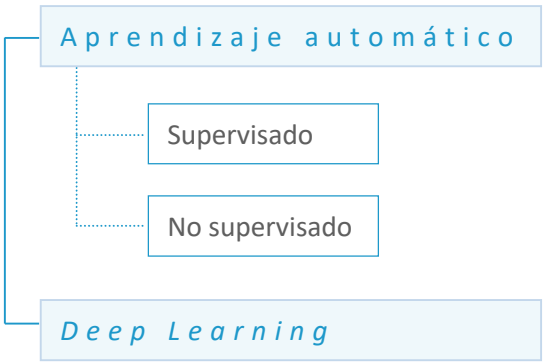


Figura 5. Aprendizaje automático.

Grandes cantidades de material hablado y escrito se pusieron a disposición de los investigadores a través de organizaciones tipo el *Linguistic Data Consortium* (LDC). Estos materiales eran fuentes de texto estándar que venían anotados con información sintáctica, semántica y pragmática. Entre estos materiales caben destacar las primeras colecciones anotadas: el *Penn Treebank*, el *Prague Dependency Treebank*, que anota la estructura de las dependencias, y las anotaciones semánticas del *PropBank*.

La existencia de estos recursos anotados promovió la tendencia de atacar los problemas más complejos del procesamiento del lenguaje natural, tipo el análisis sintáctico y semántico, como problemas de **aprendizaje automático supervisado**. Por ejemplo, se empezaron a aplicar las máquinas de vectores de soporte (SVM), el principio de máxima entropía, la regresión logística multinomial y los modelos bayesianos en la lingüística computacional. Entonces, este mayor enfoque en el aprendizaje automático llevó a una interacción más seria de los lingüistas computacionales con la comunidad estadística.

El coste y la dificultad de producir corpus anotados se convirtió en un factor limitante del uso de los enfoques supervisados para muchos problemas del procesamiento del lenguaje. Es por eso por lo que a partir de 2005 aparece una nueva tendencia hacia el uso de técnicas de **aprendizaje no supervisado** en el procesamiento del lenguaje natural. Entonces se empezaron a construir algunas aplicaciones lingüísticas a partir

de datos sin anotación alguna, por ejemplo, para la traducción automática o para el modelado de temas.

Los algoritmos de aprendizaje no supervisado se han usado en el etiquetado morfosintáctico (POS *tagging*) para agrupar palabras en las correspondientes partes del lenguaje (Goldwater y Griffiths, 2007) (Sirts, Eisenstein, Elsner y Goldwater, 2014). Además, las técnicas del aprendizaje no supervisado se han usado también para el etiquetado semántico, donde se han creado conjuntos de roles semánticos a partir de las características sintácticas (Titov y Klementiev, 2012) (Lang y Lapata, 2014).

En 2006 Geoffrey Hinton acuña el término ***Deep Learning*** (aprendizaje profundo). Con el auge de este tipo de redes neuronales en la década de 2010, estas redes de neuronas artificiales profundas se empezaron a usar en diferentes ámbitos del procesamiento del lenguaje natural. Las redes neuronales recurrentes se están utilizando como una alternativa a los modelos ocultos de Markov (HMM) en análisis morfosintáctico y en el análisis sintáctico (Chen y Manning, 2014) (Dozat, Qi, y Manning, 2017). Además, las redes neuronales profundas también se están utilizando para el etiquetado semántico (Collobert et al., 2011) (Foland Jr. y Martin, 2015). De hecho, el *Deep learning* es la base de los modelos de secuencia a secuencia (seq2seq) que se utilizan en los agentes conversacionales y *chatbots* actuales.

Accede a los ejercicios de autoevaluación a través del aula virtual

1.4. Aplicaciones del PLN

Accede al vídeo «Aplicaciones del procesamiento de lenguaje natural» a través del aula virtual

El ámbito del procesamiento del lenguaje natural tiene infinidad de aplicaciones: los agentes conversacionales (o sistemas de diálogo) que se van a estudiar en

profundidad en capítulos posteriores de esta asignatura, la traducción automática, la búsqueda de respuestas o la corrección ortográfica y la verificación gramatical.

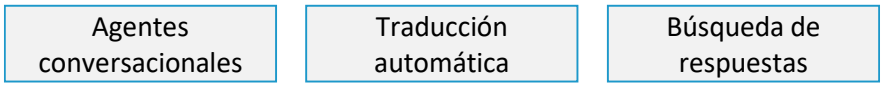


Figura 6. Aplicaciones del procesamiento del lenguaje natural.

Agentes conversacionales

También llamados sistemas de diálogo, son programas que conversan con las personas a través del lenguaje natural. Los *chatbots* son uno de los tipos más avanzados de los agentes conversacionales porque permiten mantener **conversaciones no estructuradas**, una característica de las conversaciones entre personas. Los agentes conversacionales pueden interactuar con el humano ya sea a través de la voz (hablando con el usuario), de texto, en el caso que la conversación se lleve a cabo a través de un chat, o utilizando ambas modalidades a la vez.

Los agentes conversacionales se caracterizan por ser sistemas que toman **turnos para conversar**, por lo que aparte de tener que analizar el lenguaje natural durante la conversación, deben tener en cuenta el turno de palabra. Por lo tanto, los agentes conversacionales además de tratar con tareas básicas del procesamiento del lenguaje natural como son el reconocimiento de palabras y frases o la semántica de las mismas, deben mantener el estado de la conversación y ser capaces de generar nuevas frases que continúen la conversación que mantienen con la persona.

Traducción automática

Es otra tarea relacionada con el procesamiento del lenguaje natural. El objetivo de la traducción automática es traducir automáticamente un documento de un idioma a otro. Además, se incluye en este ámbito no solo la traducción de documentos de texto, también la traducción de forma automática del habla de un lenguaje o idioma a otro.

Los mejores resultados se obtienen utilizando **métodos estocásticos basados en frases**. En estos métodos se utiliza básicamente el aprendizaje automático para analizar grandes conjuntos de datos y realizar traducciones que no contemplen las cuestiones gramaticales. En esta se requieren también herramientas para solventar la ambigüedad de las palabras como serían los **algoritmos de desambiguación**.

Búsqueda de respuestas

En inglés, ***Question Answering*** **(QA).** Es un tipo de recuperación de la información basado en el lenguaje natural. Por lo tanto, es una extensión de una búsqueda simple de información en la web, pero en lugar de solo escribir palabras clave, un usuario puede hacer preguntas completas. Los motores de búsqueda pueden responder preguntas sobre fechas y ubicaciones sin necesidad de aplicar procesamiento del lenguaje natural. Sin embargo, para responder a preguntas más complejas se requiere alguno de estos aspectos:

- La extracción de información o de un fragmento de texto en una página web.
- Hacer inferencia, es decir, sacar conclusiones basadas en hechos conocidos.
- Sintetizar y resumir información de múltiples fuentes o páginas web.

Los sistemas de búsqueda de respuestas, que requieren la comprensión de la información, se componen de diferentes elementos tales como un módulo de extracción de información, un módulo para resumir de forma automática o un módulo de desambiguación del sentido de las palabras.

Accede a los ejercicios de autoevaluación a través del aula virtual

1.5. Conocimiento del lenguaje utilizado en el PLN

Accede al vídeo «Conocimiento del lenguaje usado en PLN» a través del aula virtual

Lo que distingue a las aplicaciones de procesamiento de lenguaje de otros sistemas de procesamiento de datos es su uso del conocimiento del lenguaje. Por ejemplo, un programa que cuenta el número de bytes, palabras y líneas en un archivo de texto podemos considerarlo como una aplicación de procesamiento de datos ordinaria. Sin embargo, si para contar las palabras en el archivo de texto se requiere conocimiento sobre lo que significa ser una palabra porque se quieren contar el número de pronombres que aparecen en el archivo, ese programa se convierte en un sistema de procesamiento de lenguaje natural.

Por supuesto, este sistema de procesamiento de lenguaje es extremadamente simple en comparación con los agentes conversacionales, los sistemas de traducción automática y los sistemas de búsqueda de respuestas, que requieren un conocimiento mucho más amplio y profundo del lenguaje.

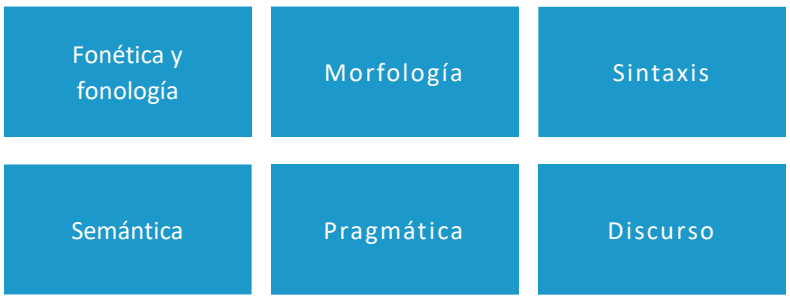


Figura 7. Conocimientos necesarios para tareas complejas de PLN.

Un agente conversacional necesita reconocer las palabras de una señal de audio y generar una señal de audio de una secuencia de palabras. Estas tareas de reconocimiento de la voz y del habla son tareas de síntesis que requieren conocimiento sobre **fonética** y **fonología**. Es decir, conocimiento de cómo se

pronuncian las palabras a partir de la secuencia de sonidos y cómo se generan cada uno de estos sonidos acústicamente.

Fonética y fonología: conocimiento sobre los sonidos lingüísticos.

El agente conversacional también necesita reconocer y saber producir variaciones de las palabras, por ejemplo, reconocer que «puertas» es el plural de una palabra y saber generar una frase en la que esta palabra se utilice en plural. Entonces, estas tareas requieren conocimiento sobre **morfología**, es decir, la forma en que las palabras se descomponen en partes que tienen un significado como puede ser la raíz de la palabra y una terminación que indique que es un plural.

Morfología: conocimiento de los componentes significativos de las palabras.

Si vamos más allá de las palabras como elementos aislados y entendidas de forma individual, un agente conversacional debe usar conocimiento estructural para encadenar las palabras que constituyen su respuesta. El agente debe saber identificar que una secuencia de palabras no tiene sentido a pesar de que el conjunto de palabras original pudiera tener sentido si se ordenaran las palabras de otra forma. El conocimiento necesario para ordenar y agrupar palabras se llama **sintaxis**.

Sintaxis: conocimiento de las relaciones estructurales entre palabras.

Para responder a una pregunta, el agente conversacional puede necesitar saber algo sobre la **semántica léxica**, es decir, sobre el significado de cada una de las palabras, así como sobre **semántica composicional** o el significado de varias palabras que se utilizan de forma conjunta en una combinación de palabras.

Semántica: conocimiento del significado.

Además de comprender el significado de las palabras, el agente conversacional puede necesitar saber algo sobre la relación de las palabras con la estructura sintáctica: si

un sintagma es un complemento circunstancial de tiempo o un complemento del nombre. Por lo tanto, el conocimiento sobre la sintaxis y el conocimiento sobre la semántica se van a utilizar de forma conjunta en el procesado del lenguaje natural.

El agente conversacional necesita determinar también el tipo de expresión que le ha interpuesto el usuario. Necesita saber si la expresión con la que este le acaba de interpelar es una pregunta a la que debe dar una respuesta hablada, una solicitud para que realice una acción o un simple enunciado o declaración sobre un hecho. Además, el agente puede determinar usar expresiones más formales y educadas en función de su interlocutor y cómo este le haya preguntado. Entonces, el agente necesita conocimiento sobre el **diálogo** o la **pragmática** para poder identificar la intención que tiene el usuario al interpelarle y dar una respuesta acorde.

Pragmática: conocimiento de la relación del significado con los objetivos y las intenciones.

Por último, el agente conversacional necesita interpretar palabras o expresiones que hacen referencia a términos que han aparecido anteriormente en la conversación. Sería el caso de pronombres o sintagmas nominales con determinantes que se refieren a partes previas del discurso. Es por eso que el agente examina las preguntas anteriores que se formularon previamente y utiliza el conocimiento sobre el **discurso** previo para resolver las referencias cruzadas.

Discurso: conocimiento sobre unidades lingüísticas más grandes que un solo enunciado.

En conclusión, para realizar tareas complejas de PLN se necesitan diferentes tipos de conocimiento del lenguaje, concretamente conocimiento sobre la fonética y fonología, la morfología, la sintaxis, la semántica, la pragmática y el discurso.

Accede a los ejercicios de autoevaluación a través del aula virtual

1.6. Referencias bibliográficas

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurich ACMGAMM Conference. *Information Processing: Proceedings of the International Conference on Information Processing*, 125-132.

Bresnan, J. y Kaplan, R. M. (1982). Introduction: Grammars as mental representations of language. En Bresnan, J. (Ed.), *The Mental Representation of Grammatical Relations*. Cambridge, Estados Unidos: MIT Press.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 740-750). Doha, Catar: Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124.

Chomsky, N. (1959). A review of B. F. Skinner's «Verbal Behavior». *Language*, 35, 26-58.

Church, K. W. (1980). *On memory limitations in natural language processing* (Tesis de maestría, Massachussetts Institute of Technology). Recuperado de https://www.researchgate.net/publication/230875927_On_Memory_Limitations_in_Natural_Language_Processing

Cohen, P. R. y Perrault, C. R. (1979). Elements of a planbased theory of speech acts. *Cognitive Science, 3*(3), 177–212.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. y Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12, 2493–2537.

Colmerauer, A. (1978). Metamorphosis grammars. En L. Bolc (Ed.), *Natural Language Communication with Computers, Lecture Notes in Computer Science 63* (pp. 133-189). Berlín, Alemania: Springer Verlag.

Davis, K. H., Biddulph, R. y Balashek, S. (1952). Automatic recognition of spoken digits. *Journal of the Acoustical Society of America, 24*(6), 637-642.

Dozat, T., Qi, P., y Manning, C. D. (2017). Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. En *Proceedings of the CoNLL 2017 Shared Task* (pp. 20-30). Vancouver, Canadá: Association for Computational Linguistics.

Fillmore, C. J. (1968). The case for case. En E. W. Bach y R. T. Harms, (Eds.), *Universals in Linguistic Theory* (pp. 1-88). Nueva York, Estados Unidos: Holt, Rinehart & Winston.

Foland Jr., W. R. y Martin, J. H. (2015). Dependency-based semantic role labeling using convolutional neural networks. En *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)* (pp. 279-289). Denver, Estados Unidos: *SEM 2015 Organizing Committee.

Goldwater, S. y Griffiths, T. L. (2007). A fully Bayesian approach to unsupervised part-of-speech tagging. *Annual Meeting of the Association of Computational Linguistics ACL-07*, 744-751. Recuperado de http://aclweb.org/anthology/P07-1094

Grosz, B. J. (1977). The representation and use of focus in a system for understanding dialogs. *International Joint Conference on Artificial Intelligence*, 67-76.

Grosz, B. J. y Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics, 12*(3), 175-204.

Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, 44, 311-338.

Hobbs, J. R. (1979). Coherence and coreference. *Cognitive Science*, 3, 67-90.

Kaplan, R. M. y Kay, M. (1981). Phonological rules and finite-state transducers. *Annual meeting of the Linguistics Society of America*. New York, Estados Unidos: Linguistic Society of America.

Kay, M. (1979). *Functional grammar*. *Proceedings of the Annual Meeting of the Berkeley Linguistics Society, Estados Unidos*. Recuperado de http://linguistics.berkeley.edu/bls/previous_proceedings/bls5.pdf

Kleene, S. C. (1951). *Representation of events in nerve nets and finite automata*. Santa Mónica, Estados Unidos: RAND Corporation. Recuperado de https://www.rand.org/pubs/research_memoranda/RM704.html

Kucera, H. y Francis, W. N. (1967). *Computational analysis of present-day American English*. Providence, Estados Unidos: Brown University Press.

Lang, J. y Lapata, M. (2014). Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics, 40*(3), 633-669.

McCulloch, W. S. y Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115-133.

Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijnagaarden, A. y Woodger, M. (1960). Report on the algorithmic language ALGOL 60. *Communications of the ACM, 3*(5), 299–314.

Newell, A., Shaw, J. C. y Simon, H. A. (1959). Report on a general problem-solving program. En UNESCO (Ed.), *Information processing: proceedings of the International*

Conference on Information Processing (pp. 256-264). Múnich, Alemania: Oldenbourg Verlag.

Pereira, F. C. N. y Warren, D. H. D. (1980). Definite clause grammars for language analysis: a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence, 13*(3), 231-278.

Perrault, C. R. and Allen, J. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics, 6*(3-4), 167-182.

Quillian, M. R. (1968). Semantic memory. En M. Minsky (Ed.), *Semantic Information Processing* (pp. 227–270). Cambridge, Estados Unidos: MIT Press.

Schank, R. C. y Riesbeck, C. K. (Eds.). (1981). *Inside computer understanding: five programs plus miniatures*. New Jersey, Estados Unidos: Lawrence Erlbaum.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*(3), 379-423.

Simmons, R. F. (1973). Semantic networks: Their computation and use for understanding English sentences. En R. C. Schank y K. M. Colby (Eds.), *Computer Models of Thought and Language* (pp. 61-113). San Francisco, Estados Unidos: W.H. Freeman and Co.

Sirts, K., Eisenstein, J., Elsner, M., y Goldwater, S. (2014). POS induction with distributional and morphological information using a distance-dependent Chinese restaurant process. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics,* 265-271. Recuperado de http://www.aclweb.org/anthology/P14-2044

Titov, I. and Klementiev, A. (2012). A Bayesian approach to unsupervised semantic role induction. *Proceedings of the 13th Conference of the European Chapter of the*

Association for Computational Linguistics, 12–22. Recuperado de https://dl.acm.org/citation.cfm?id=2380821

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*(1), 230-265.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology, 3*(1), 1-191.

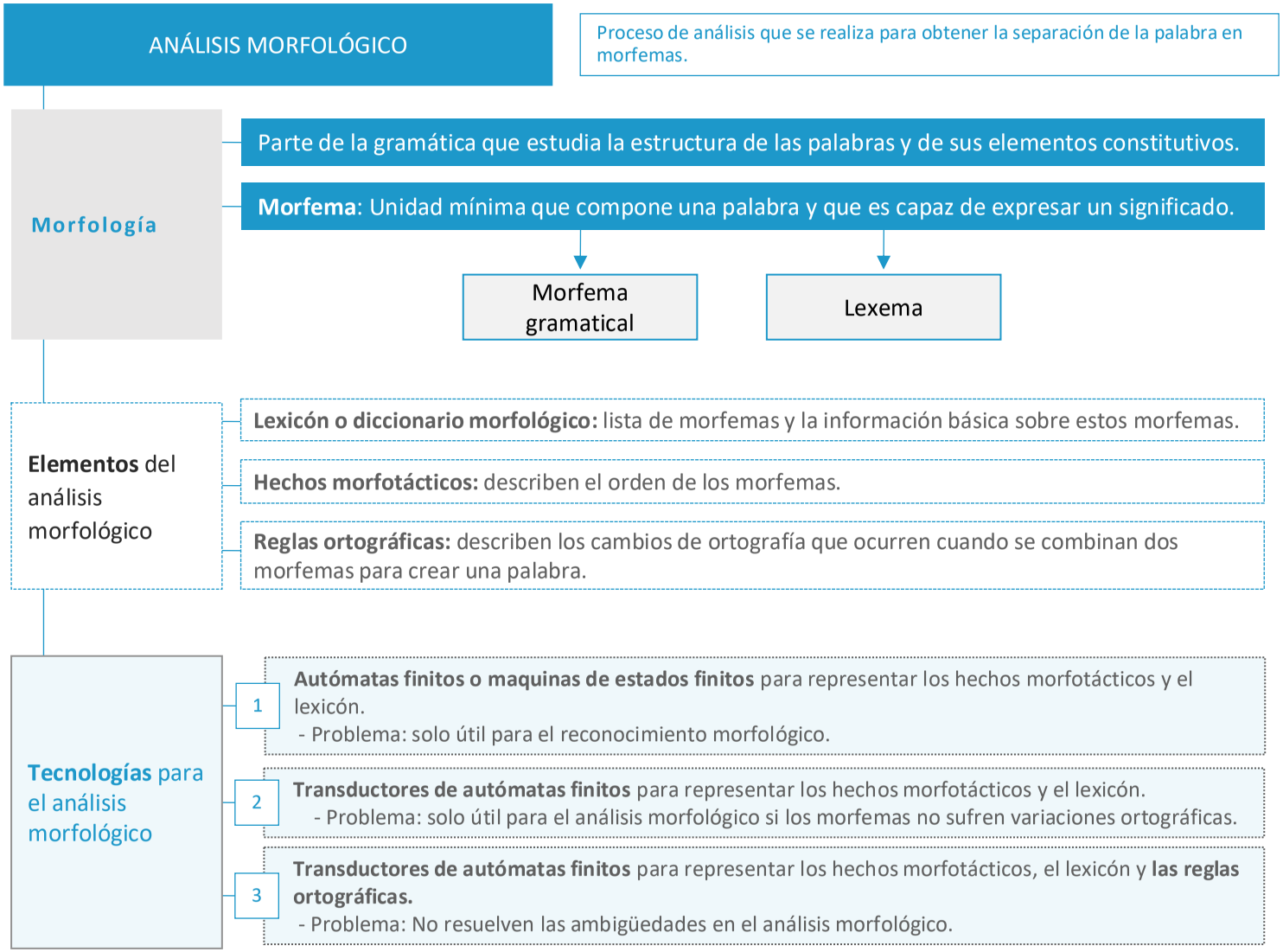
Woods, W. A. (1967). *Semantics for a Question-Answering System*. Nueva York, Estados Unidos: Garland Publishing.

Accede al vídeo «Resumen» a través del aula virtual

Tema 2. Análisis morfológico

Índice

Esquema



Ideas clave

2.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Este tema comienza definiendo lo que es la morfología. Después se describen los elementos involucrados en el análisis morfológico (lexicón, hechos morfotácticos y reglas ortográficas) y que son imprescindibles para realizar este análisis de forma automática.

También se describe cómo representar una versión simple de un lexicón que sirve para el reconocimiento morfológico y que incluye el conocimiento morfotáctico modelándolo como una máquina de estados finitos. Seguidamente, se presenta los transductores de autómatas finitos como una forma de modelar las características morfológicas en el lexicón y de abordar el análisis morfológico como la traducción de una cadena de caracteres a una cadena de morfemas.

Finalmente, se muestra cómo usar los transductores de autómatas finitos para modelar las reglas ortográficas necesarias en al análisis morfológico si se dan cambios de ortografía a la hora de combinar varios morfemas para crear una palabra. Entonces, estos transductores que modelan las reglas ortográficas se combinan con el transductor que recoge la información léxica para poder realizar el análisis morfológico.

Los apartados de los que consta este tema son:

- Morfología.
- Elementos del análisis morfológico.

- Uso de autómatas finitos para el reconocimiento morfológico.
- Análisis morfológico basado en transductores de autómatas finitos.
- Análisis morfológico utilizando un lexicón y reglas ortográficas representados como transductores de autómatas finitos.

Al finalizar el estudio de este tema serás capaz de:

- Describir los conceptos fundamentales de la morfología en español.
- Identificar los diferentes elementos necesarios para realizar el análisis morfológico de forma automática.
- Entender el posible uso de autómatas finitos en el reconocimiento morfológico y aplicarlos para crear un lexicón.
- Aplicar los transductores de autómatas finitos para abordar el análisis morfológico como la traducción de una cadena de caracteres a una cadena de morfemas.
- Representar las reglas ortográficas como transductores de autómatas finitos y utilizarlos junto con el lexicón para realizar el análisis morfológico.

Accede a los ejercicios de autoevaluación a través del aula virtual

2.2. Morfología

Accede al vídeo «Morfología» a través del aula virtual

La Real Academia Española (RAE) en su diccionario de la lengua española define la palabra **morfología:** «De *morfo-* y *-logía*. 2. f. Gram. Parte de la gramática que estudia la estructura de las palabras y de sus elementos constitutivos».

Por tanto, la morfología estudia la forma en que las palabras se descomponen en partes indivisibles, las cuales tienen un significado. A estas unidades mínimas se les

llama **morfemas** y, por ejemplo, la palabra «mujeres» contiene dos morfemas: el primero es «mujer» y el segundo es «-es».

Un **morfema** es la unidad mínima que compone una palabra y que es capaz de expresar un significado.

De hecho, no se debe confundir esta definición de morfema con la de morfema gramatical, al que en algunos textos se le llama de la misma manera. Se entiende como **morfema gramatical** al que tiene un significado gramatical, es decir, significado sobre:

- El género (masculino o femenino).
- Número (singular o plural).
- Persona (p. ej. tercera persona del singular).
- Modo y tiempo (p. ej. modo indicativo y tiempo futuro).

En contraposición al morfema gramatical está el lexema. Un **lexema** sería el morfema que conforma la raíz o parte invariante de la palabra y que es la mínima unidad con significado léxico, por lo que proporciona el significado principal de la palabra.

Para el ejemplo de la palabra «mujeres», el morfema «mujer» sería el lexema o raíz de la palabra con significado léxico y el morfema «-es» sería el morfema gramatical que añade significado gramatical y expresa el número plural. El singular, que también sería en este caso «mujer», contiene a su vez dos morfemas: el lexema «mujer» y un morfema cero de número singular. Un **morfema cero** es un morfema gramatical sin realización fonética, pero que tiene significado gramatical.

Una palabra como «cantábamos» contiene tres morfemas:

- El primero es el lexema «cant», que indica el significado léxico de la palabra, es decir, el acto de que una persona produzca con la voz sonidos melodiosos.
- El segundo es el morfema «-aba», que proporciona el significado gramatical de modo indicativo y tiempo pasado.

- El tercero es el morfema «-mos», que indica el significado gramatical de primera persona del plural.

En el procesamiento del lenguaje natural, el ámbito de la morfología computacional trata de reconocer de forma automática los morfemas que contiene una palabra.

El **análisis morfológico** es el proceso de análisis que se realiza para obtener la separación de la palabra en morfemas.

Conocer la morfología de las palabras es importante para, si se está buscando en un texto el verbo «pensar», que se reconozca también la fórmula «piénsalo», aunque ambas no se compongan de la misma cadena de caracteres en la raíz. Además, a la hora de generar de forma automática frases o expresiones de texto se debe conocer la morfología de las palabras a utilizar para encajar, por ejemplo, el género y el número de un nombre con el adjetivo que la acompaña o la persona del verbo con el sujeto de la frase.

El análisis morfológico en inglés, lengua en la que se realiza la mayoría de la investigación en el ámbito de la morfología computacional, es más simple que el análisis morfológico en español. Esto se debe principalmente a que en el español se dan muchas variedades diferentes de las formas en las terminaciones de las palabras y también se dan más alteraciones en la raíz de las mismas.

El uso de transductores de autómatas finitos, que se explica en las siguientes secciones, es la forma más habitual de realizar el análisis morfológico. Sin embargo, hay otras formas más simples como sería usar el método de extracción de raíces (*stemming*) de Porter (1980) que aplica reglas en cascada para cortar de forma abrupta los morfemas eliminando las terminaciones y así obtener las raíces de las palabras.

Accede a los ejercicios de autoevaluación a través del aula virtual

2.3. Elementos del análisis morfológico

Accede al vídeo «Tecnologías para el análisis morfológico» a través del aula virtual

El análisis morfológico pretende separar una palabra en morfemas. En el análisis morfológico automático se va a obtener como salida la raíz de la palabra, también llamada **lema**, y la información gramatical que aportan los diferentes morfemas, llamada **características morfológicas**.

La tabla 1 muestra un ejemplo de la salida de un analizador morfológico para varias palabras de entrada donde se puede apreciar la raíz de la palabra o lema y las características morfológicas de estas. Por ejemplo, para la primera entrada de la palabra «vino» el lema es «venir» y las características morfológicas de esta palabra son que es un verbo (+V), que el tiempo es el pasado (+Perf) y que la persona es la tercera persona del singular (+3P +Sg). Para la segunda entrada de la palabra «vino», el lema es «vino» y las características morfológicas de esta palabra son que es un nombre (+N) de género masculino (+Masc) y número singular (+Sg). Tal como se acaba de observar, una misma entrada puede devolver diferentes resultados en el análisis morfológico, por lo tanto, será necesario tratar la **ambigüedad** tal como se va a ver en temas posteriores.

Entrada	Análisis Morfológico
lugar	lugar +N +Masc +Sg
vino	venir +V +Perf +3P +Sg
vino	vino +N +Masc +Sg
vinos	vino +N +Masc +Pl
bebo	beber +V +PInd +1P +Sg

Tabla 2. Ejemplo de la salida de un analizador morfológico.

Para construir un sistema capaz de realizar el análisis morfológico de forma automática son necesarios tres elementos: un lexicón, un conjunto de hechos morfotácticos y un conjunto de reglas ortográficas.

El **lexicón** es un catálogo de las palabras de una lengua, por lo tanto, este debería contener todas y cada una de las palabras de una lengua. Como esto es inviable, lo que se hace es crear un diccionario o base de datos que contiene información básica sobre los morfemas y, además, generar un conjunto de hechos morfotácticos que explican cómo estos morfemas se pueden concatenar para formar palabras. Entonces el lexicón solo contiene información sobre los morfemas, por ejemplo, si la raíz de una palabra es un nombre o un verbo.

Un lexicón o diccionario morfológico es una lista de morfemas y la información básica sobre estos morfemas.

Los **hechos morfotácticos** hacen referencia al modelo que describe el orden de los morfemas y a partir del cual es posible generar todas las palabras de un léxico. Por ejemplo, uno de estos hechos es que en español los morfemas que expresan el número plural solo pueden ir concatenados detrás del nombre, pero nunca delante de este como pasaría en otros idiomas. Por tanto, la definición formal de este hecho sería lo que se llama un hecho morfotáctico.

Los hechos morfotácticos describen el orden de los morfemas, es decir, por qué unas clases de morfemas pueden seguir a otras clases de morfemas dentro de las palabras.

Por último, las **reglas ortográficas** se usan para modelar los cambios de ortografía que ocurren en una palabra cuando se concatenan dos morfemas. Un caso sería que, en español, los sustantivos de origen extranjero terminados en «-y» forman su plural añadiendo el morfema «-s», pero deben cambiar la terminación de la raíz «y» por una «i». Un ejemplo concreto que cumple esta regla ortográfica sería la palabra «espray» cuya fórmula en plural sería «espráis».

Las reglas ortográficas describen los cambios de ortografía que ocurren cuando se combinan dos morfemas para crear una palabra.

En el siguiente apartado se describe, tal y como indicábamos al inicio de este tema, cómo representar una versión simple de un lexicón, que sirve para el reconocimiento morfológico, y que incluye el conocimiento morfotáctico modelándolo como una máquina de estados finitos. Seguidamente, se presentan los transductores de autómatas finitos como una forma de modelar las características morfológicas en el lexicón y de abordar el análisis morfológico como la traducción de una cadena de caracteres a una cadena de morfemas. Finalmente, se muestra cómo usar los transductores de autómatas finitos para modelar las reglas ortográficas necesarias en al análisis morfológico si se dan cambios de ortografía a la hora de combinar varios morfemas para crear una palabra. Entonces, estos transductores que modelan las reglas ortográficas se combinan con el transductor que recoge la información léxica para poder realizar el análisis morfológico.

Accede a los ejercicios de autoevaluación a través del aula virtual

2.4. Uso de autómatas finitos para el reconocimiento morfológico

Accede al vídeo «Tecnologías para el análisis morfológico» a través del aula virtual

Un lexicón que sirva para el análisis morfológico debe contener una lista de morfemas, además del conocimiento morfotáctico, para combinar o concatenar dichos morfemas. La forma más común de modelar los hechos morfotácticos en una versión simple de un lexicón es utilizando autómatas finitos, también llamados máquinas de estados finitos.

Una **máquina de estados finitos** o **autómata finito** es un modelo computacional que permite producir de forma automática una salida a partir de una entrada con base en una serie de transiciones entre estados.

En las máquinas de estados finitos, los estados se representan como vértices de un grafo. Una transición desde un estado a otro se representa mediante una arista que se dirige desde un nodo al otro nodo y la arista se etiqueta con el nombre del símbolo que hace que se cambie de un estado al otro.

Existe la posibilidad de que no sea necesaria ninguna entrada para que se dé una transición entre dos estados, es lo que se llama una **transición vacía o transición** $\in$, y se marca la arista correspondiente con el símbolo $\in$. El estado inicial de la máquina de estados se denomina q_0 y los estados finales se pueden distinguir porque los vértices que los representan tienen una doble circunferencia.

Ejemplo ilustrativo 1. Conocimiento morfotáctico de la formación de plurales de los nombres en inglés, modelado como una máquina de estados finitos.

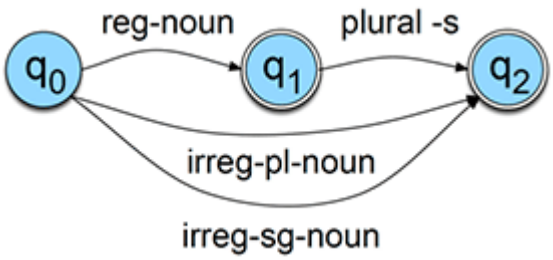


Figura 8. Máquina de estados finitos.
Fuente: Jurafsky y Martin, 2009.

La figura 1 muestra un ejemplo muy sencillo de máquina de estados finitos que recoge el conocimiento morfotáctico de cómo se forman los plurales de los nombres en inglés.

Si seguimos el ejemplo ilustrativo 1, en el caso de que el lexicón contenga nombres regulares (*reg-noun*), el plural se crea añadiendo el morfema «-s». Un ejemplo de este caso sería el nombre *«cat»* (gato) cuyo plural es *«cats»* (gatos). Se observa que

a partir del estado inicial q_0, si se da el caso de que un nombre es regular, se sigue la transición al estado q_1 y luego, para hacer el plural, se cambia al estado q_2 añadiendo la terminación «-s».

Sin embargo, si el nombre que contiene el lexicón fuera irregular (*irreg-sg-noun* o *irreg-pl-noun*) se pasaría directamente del estado q_0 al estado q_2. Sería el caso del nombre irregular con número singular (*irreg-sg-noun*), *«goose»* (ganso), en el que se obtendría el plural *«geese»* (gansos).

El conocimiento morfotáctico para un lexicón se modela utilizando máquinas de estados finitos que recogen las diferentes derivaciones que se pueden hacer de los lemas contenidos en el lexicón por el simple hecho de ir combinando morfemas.

Con una máquina de estados finitos similar a la presentada en el ejemplo ilustrativo 1, se podría modelar la creación de las fórmulas de plural en español que se generan concatenando el morfema «-s» al singular de los nombres. Otras máquinas de estados servirían para generar las diferentes formas verbales a partir de a raíz de los verbos y las diferentes terminaciones verbales para cada persona, modo y tiempo, es decir, a partir de los morfemas recogidos en el lexicón.

Las máquinas de estados finitos que modelan el conocimiento morfotáctico se pueden usar en el **reconocimiento morfológico** para determinar si una cadena de caracteres de entrada constituye una palabra correcta o no.

En el reconocimiento morfológico, lo que se hace es conectar cada uno de los lemas recogidos en el lexicón a la máquina de estados finitos que modela el conocimiento morfotáctico. Es decir, que se expanden cada uno de los arcos en la máquina de estados finitos con todos los morfemas que conforman el conjunto de raíces. Entonces, la máquina de estados resultante se define a nivel de letras y permite reconocer palabras correctas empezando desde el estado inicial y recorriendo los estados con base en la cadena de letras que componen la entrada.

Ejemplo ilustrativo 2. Máquina de estados para el reconocimiento morfológico de plurales de los nombres en inglés «*fox*» (zorro), «*cat*» (gato) y «*goose*» (ganso).

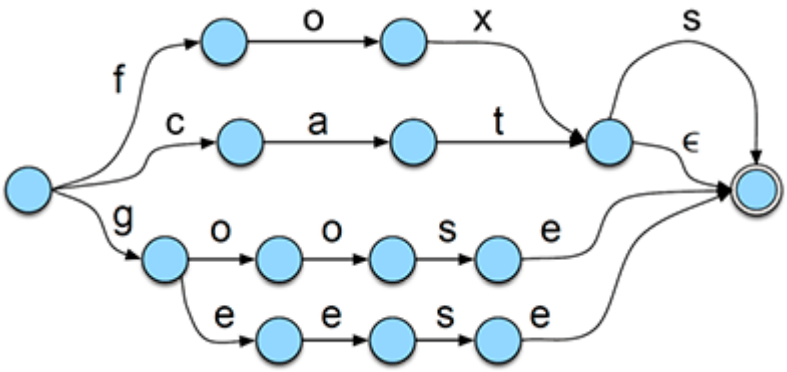


Figura 9. Expansión de la máquina de estados finitos presentada en el ejemplo ilustrativo 1 y que modela la formación de plurales de los nombres en inglés recogidos el lexicón de la tabla 2.
Fuente: Jurafsky y Martin, 2009.

En el proceso de reconocimiento morfológico, la máquina de estados finitos que se presenta en la figura 2 sirve para determinar si una cadena de caracteres de entrada constituye una palabra correcta o no, concretamente permite reconocer palabras que son plurales de nombres en inglés. La creación de esta máquina de estados finitos requiere conectar cada uno de los morfemas recogidos en el lexicón presentado en la tabla 2 a la máquina de estados finitos que modela el conocimiento morfotáctico presentado en la figura 1 del ejemplo ilustrativo 1.

En este proceso, el arco *reg-noun* se expande para cada uno de los morfemas del tipo *reg-noun* que aparecen en el lexicón, es decir para «*fox*» y «*cat*». Por lo tanto, se remplaza el arco *reg-noun* con una secuencia de estados en la cual cada letra del morfema representa una transición y se modela con un nuevo arco. Como se observa, la máquina de estados resultante se compone de transiciones que hacen referencia a las letras que conforman las palabras de número plural derivadas de la definición del lexicón y del conocimiento morfotáctico para este ejemplo.

reg-noun	irreg-pl-noun	irreg-sg-noun	plural
fox	geese	goose	-s

cat			

Tabla 3. Lexicón que incluye los nombres regulares (*reg-noun*), el nombre irregular con número singular (*irreg-sg-noun*) y el nombre irregular con número plural (*irreg-pl-noun*).

Accede a los ejercicios de autoevaluación a través del aula virtual

2.5. Análisis morfológico basado en transductores de autómatas finitos

Accede al vídeo «Análisis morfológico» a través del aula virtual

En la sección anterior veíamos cómo las máquinas de estados finitos, también llamados autómatas finitos, permiten modelar el conocimiento morfotáctico de un lexicón y pueden ser utilizados para reconocer las palabras correctas que se derivan del lexicón. Sin embargo, para el análisis morfológico se necesita traducir la cadena de letras que componen la palabra generada a partir del conocimiento morfotáctico del lexicón en una cadena de morfemas. Esta funcionalidad de traducción que toma como entrada una cadena de letras y produce como resultado una cadena de morfemas la pueden realizar los **transductores de autómatas finitos**.

Un transductor de autómatas finitos es un tipo de autómata finito que mapea dos conjuntos de símbolos y para ello se compone de dos cintas, una de entrada y otra de salida.

Estos transductores se pueden ver como autómatas finitos con dos cintas que permiten reconocer o generar dos pares de cadenas de caracteres (una por cada una de las cintas) o, lo que es lo mismo, traducir desde la cadena de símbolos de la cinta de entrada a la cadena de símbolos de la cinta de salida. Así, en el análisis morfológico, el transductor de autómatas finitos tiene una función más general que

el autómata finito y es capaz de traducir la cadena de letras que componen la palabra a una cadena de morfemas, incluyendo también sus características morfológicas.

Por ejemplo, se traduciría la cadena de letras de la entrada «vinos» a la salida morfológica «vino +N +Masc +Pl», lo que indicaría el lema o la raíz de la palabra «vino», además de que esta palabra es un nombre (+N), de género masculino (+Masc) y número plural (+Pl).

En el paradigma de la **morfología de estados finitos:**

- Una palabra se modela como la correspondencia entre un **nivel léxico**, que representa la concatenación del lema de la palabra y las características morfológicas del resto de morfemas que forman la palabra.
- Y el **nivel de superficie**, que representa la concatenación de las letras que componen la ortografía de la palabra.

Para el ejemplo anterior, «vinos» sería el nivel de superficie y «vino +N +Masc +Pl» el nivel léxico.

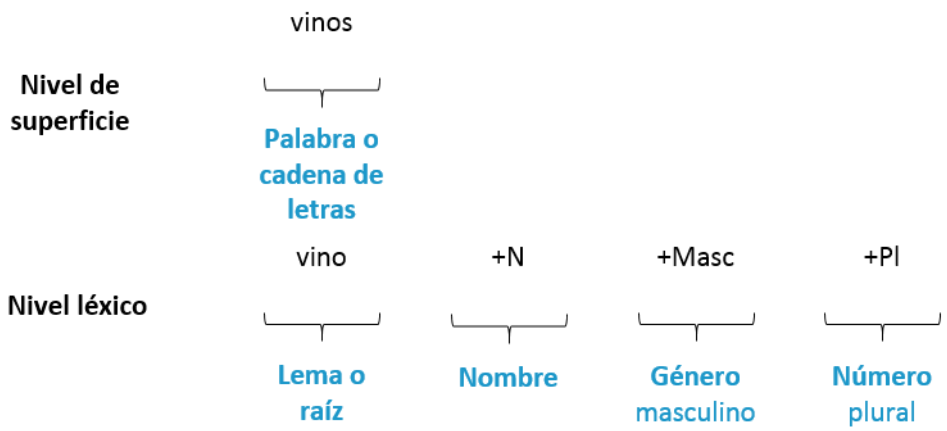


Figura 10. Análisis morfológico de «vinos».

Análisis morfológico

En el análisis morfológico, el transductor de autómatas finitos tiene:

- La cinta de léxico.
- La cinta de superficie.
- También presenta un símbolo de cada alfabeto en cada uno de los arcos.

Así, en cada arco de la máquina de estados finitos hay un **par de símbolos** $s_1 : s_2$ (léxico:superficie) que representan el mapeo entre un símbolo del nivel léxico (s_1) y otro del nivel de superficie (s_2). En el caso de que un símbolo se mapee en sí mismo, se habla de pares por defecto y puede referirse a ellos simplemente con un único símbolo s.

Un **analizador morfológico basado en transductores de autómatas finitos** se puede construir a partir de una máquina de estados finitos que modela el conocimiento morfotáctico y del lexicón, como la presentada en la sección anterior. Simplemente es necesario añadirle al autómata finito una cinta léxica extra que recoja las características morfológicas; por ejemplo, si el morfema corresponde al número singular (+Sg) o al número plural (+Pl). El lexicón, en este caso y a diferencia de cuando se tenía una única cinta en la máquina de estados finitos, también tiene dos niveles: léxico y de superficie.

El lexicón de dos niveles se utiliza para expandir el transductor de autómatas finitos con los términos que este recoge, es decir, que el lexicón sirve para remplazar los arcos genéricos en el transductor de autómatas finitos con los términos que conforman el vocabulario. Una vez creado el transductor de autómatas finitos, este permite realizar el análisis morfológico empezando desde el estado inicial, recorriendo los estados con base en la cadena de letras que componen la palabra de entrada y obteniendo como salida el lema y las diferentes características morfológicas de la palabra.

> **Ejemplo ilustrativo 3.** Transductor de autómatas finitos que permite analizar morfológicamente los nombres en inglés *"fox"* (zorro), *"cat"* (gato) y *"goose"* (ganso) tanto en sus formas de número singular como plural.

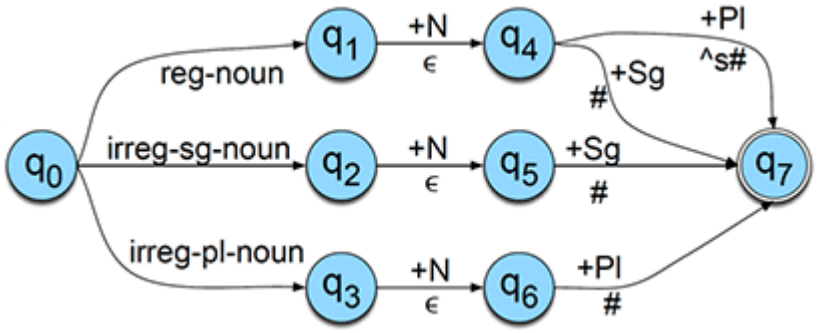


Figura 11. Transductor de autómatas finitos.
Fuente: Jurafsky y Martin, 2009.

El transductor de autómatas finitos presentado en esta imagen extiende la máquina de estados finitos que modela el conocimiento morfotáctico para la creación de plurales en inglés.

En este ejemplo ilustrativo 3, a la máquina de estados finitos de la figura 1 del ejemplo ilustrativo 1 se le ha añadido una cinta léxica para modelar las características morfológicas que hacen referencia a los nombres (+N) de número singular (+Sg) o plural (+Pl).

A partir del estado inicial q_0, si la cadena de caracteres de entrada se corresponde con un nombre regular (*reg-noun*) de los registrados en el lexicón, se sigue la transición al estado q_1. Luego, sin necesidad de que se dé ninguna entrada, esto es, en una transición vacía o transición ∈ a nivel de superficie (símbolo debajo del arco), se cambia al estado q_4. Durante esta transición, a nivel léxico se produce como salida la etiqueta +N (símbolo encima del arco) que indica que el morfema es un nombre, aportando al análisis esta característica morfológica.

Una vez en el estado q_4 , existen dos posibles transiciones que llevan al mismo estado final q_7, pero que producen diferentes salidas a nivel léxico:

- Si a la entrada se detecta la frontera entre morfemas (^) en la cinta de superficie, seguido por el carácter «s» y de la frontera entre palabras (#), a la salida se produce la etiqueta +Pl, que indica la característica morfológica de número plural.

- Por el contrario, si a la entrada se detecta la frontera entre palabras (#) en la cinta de superficie, la salida será la etiqueta *+Sg,* que indica la característica morfológica de número singular.

Un ejemplo de lexicón se presenta en la tabla 3. Este lexicón extiende el presentado en la tabla 2 del ejemplo ilustrativo 1 y añade información léxica a la información de superficie. Esto se debe a que, en el análisis morfológico, el lexicón utilizado junto con el transductor de autómatas finitos tiene dos niveles, un nivel léxico y otro de superficie. Con lo que, para cada cadena de caracteres a nivel de superficie se le asigna la correspondiente cadena de caracteres a nivel léxico, dicho de otra manera, para cada una de las palabras en el lexicón se les mapea a su correspondiente lema o raíz.

Otro caso sería el del nombre irregular con número plural *«geese»* (gansos); en el análisis morfológico se tiene que obtener la información «*goose +N +Pl»* y, por lo tanto, el lexicón tiene que mapear de la palabra «*geese»* (gansos) al lema *«goose»* (ganso). Para realizar este mapeo en el lexicón se utilizan los pares de símbolos, como sería el par «o:e», que indica que el símbolo de superficie «e» se traduce a la salida en el símbolo léxico «o».

En el caso de un par por defecto, aquel en que el símbolo se mapea en sí mismo, se observa la notación abreviada donde «g» indicaría «g:g» un símbolo «g» en la cinta de superficie se traduce en un símbolo «g» en la cinta de léxico.

reg-noun	irreg-pl-noun	irreg-sg-noun
fox	g o:e o:e s e	goose
cat		

Tabla 4. Lexicón con dos niveles, léxico y de superficie, para los nombres regulares (reg-noun) «fox» (zorro) y «cat» (gato), el nombre irregular con número singular (irreg-sg-noun) «goose» (ganso) y el nombre irregular con número plural (irreg-pl-noun) «geese» (gansos).

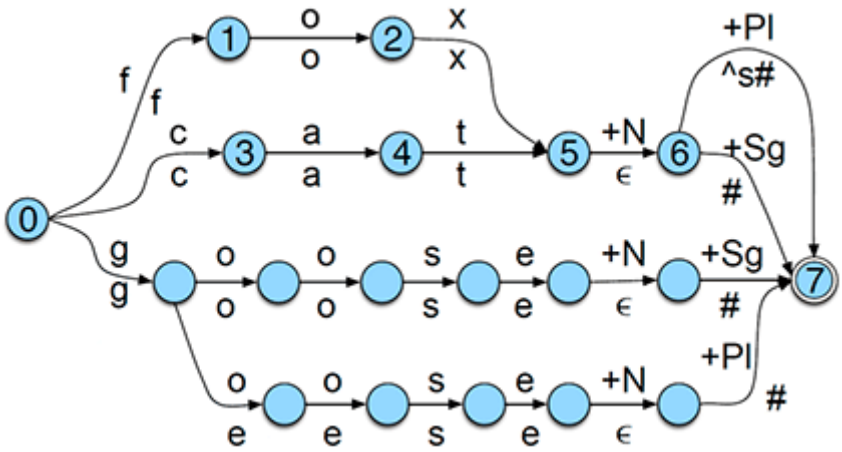


Figura 12. Expansión del transductor de autómatas finitos (figura 4) con los términos recogidos el lexicón de la tabla 3.
Fuente: Jurafsky y Martin, 2009.

El transductor de autómatas finitos de dos cintas (figura 5) se obtiene de expandir el transductor de autómatas finitos de la figura 4 con los términos recogidos en el lexicón de la tabla 3.

En la creación de este transductor de autómatas finitos el arco *reg-noun* se expande para cada uno de los términos del tipo *reg-noun* que aparecen en el lexicón, es decir para «*fox*» y «*cat*». Por lo tanto, se remplaza el arco *reg-noun* con una secuencia de estados en la cual cada transición modela con un nuevo arco las dos cintas, una que proporciona el nivel léxico y otra el nivel de superficie. Entonces la entrada en el nivel de superficie «*cat*» se mapea en «*cat* +N +Pl» a nivel léxico. Es decir, que el transductor de autómatas finitos resultante permite realizar el análisis morfológico de algunos nombres en inglés.

Accede a los ejercicios de autoevaluación a través del aula virtual

2.6. Análisis morfológico utilizando un lexicón y reglas ortográficas representados como transductores de autómatas finitos

Accede al vídeo «Análisis morfológico» a través del aula virtual

Los transductores de autómatas finitos descritos en la sección anterior pueden realizar el análisis morfológico y reconocer correctamente el lema y las características morfológicas de aquellas palabras que se forman concatenando directamente morfemas sin que estos sufran variaciones ortográficas.

Como ya hemos explicado, la palabra «vinos» se crea al concatenar el lema «vino» y el morfema de número plural «-s», por lo tanto, el transductor de autómatas finitos puede obtener la información morfológica «vino +N +Masc +Pl» al analizar la palabra (véase figura 3). Sin embargo, si a la hora de crear una palabra **se ha modificado alguno de los morfemas** que la componen, los transductores de autómatas finitos descritos en la sección anterior no funcionan correctamente.

Para generar el plural del nombre «nuez» no es suficiente con añadir el morfema «-s»; se debe intercalar la letra «e» y, además, se tiene que cambiar la letra «z» por la letra «c». Aquí, el transductor de autómatas finitos de dos cintas que recibe en la entrada la cadena de caracteres «nueces» no va a ser capaz de determinar que el lema es «nuez», ni que una de las características morfológicas es el número plural (+Pl) ni generar a la salida la cadena «nuez +N +Fem +Pl». Solo sería capaz de realizar correctamente el análisis morfológico si conociera la regla ortográfica que establece que se debe añadir una letra «e» para hacer los plurales de nombres acabados en consonante, y la otra regla que estipula que, si el nombre acaba en la letra «z», esta debe cambiarse por una «c».

Las reglas ortográficas

Las reglas ortográficas describen los cambios de ortografía que ocurren cuando se combinan dos morfemas para crear una palabra y son imprescindibles para realizar correctamente el análisis morfológico. En PLN, estas reglas ortográficas se implementan también como **transductores de estados finitos**.

Cada una de las reglas ortográficas se representa como un transductor de estados finitos capaz de producir un cambio concreto o modificación en la ortografía de la palabra resultante de concatenar los morfemas. Así, el transductor de estados finitos asociado a una regla ortográfica es capaz de traducir la cadena que contiene la simple concatenación de los morfemas en otra cadena de morfemas que incluya las variaciones necesarias para formar una palabra ortográficamente correcta.

Por ejemplo, un transductor de estados finitos podría representar la regla ortográfica del español que trata sobre hacer los plurales de nombres acabados en consonante añadiendo la letra «e». Con este transductor tendríamos la cadena de morfemas «lugar^s#»:

- Resultante de unir el lema «lugar» y el morfema de número plural «-s».
- Donde el símbolo (^) indica la frontera entre morfemas.
- El símbolo (#) indica la frontera entre palabras.

Así, se podría traducir en la cadena de caracteres «lugares» que conforma una palabra ortográficamente correcta.

> **Ejemplo ilustrativo 4.** Uso de un transductor de estados finitos para representar la regla ortográfica en inglés de que se debe añadir una «e» antes del morfema «-s» para formar el plural de los nombres que acaban en «-z», «-s» y «-x».

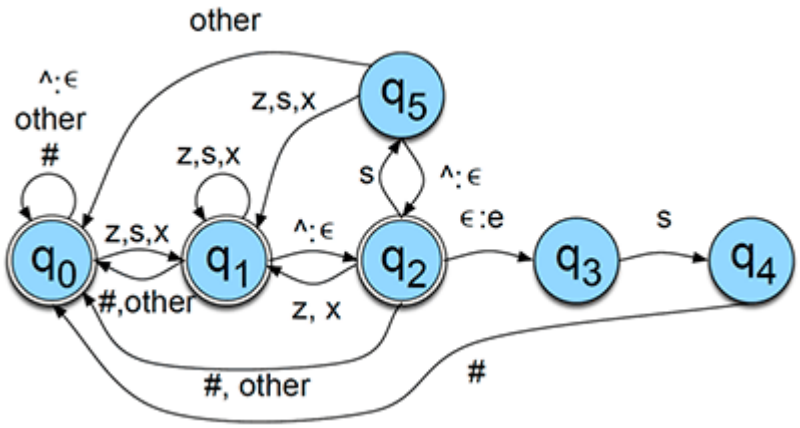


Figura 13. Transductor de estados finitos asociado a una regla ortográfica.
Fuente: Jurafsky y Martin, 2009.

El transductor de autómatas finitos presentado en la figura 4 del ejemplo ilustrativo 3 detecta que el plural del nombre *«fox»* (zorro) es *«foxs»*. Esto no es correcto porque en inglés los nombres acabados en «-z», «-s» y «-x» generan su plural añadiendo una «e» antes del morfema «-s». La fórmula correcta para el plural de *«fox»* es *«foxes»* (zorros) y, por lo tanto, es necesario tener en cuenta esta regla ortográfica a la hora de realizar el análisis morfológico.

En la figura 6 del ejemplo ilustrativo 4 se ve que:

- El estado inicial q_0 modela que se haya encontrado cualquier par de símbolos que no estén relacionados con la regla y por eso es un estado final que permite ignorar cualquier cadena que no cumpla la regla.

- Los estados q_1 y q_2 también son aceptados como estados finales:
  - El estado q_1 modela haberse encontrado alguno de los siguientes caracteres: «-z», «-s» y «-x».
  - El estado q_2 modela haberse encontrando la frontera entre morfemas (^) después del carácter «-z», «-s» o «-x».
- El estado q_3 modela haberse encontrado con la inserción de una «e», pero no es un estado final porque es necesario que a la «e» le siga el morfema «-s», representado en la transición del estado q_3 al estado q_4, y después se encuentre el final de palabra (#) en la transición del estado q_4 al estado q_0.

- El estado q_5 se usa para garantizar que siempre se inserta una «e» cuando se ha encontrado una «s» después de la frontera entre morfemas.

- Por último, el símbolo (other) se utiliza en la figura 6 para marcar las transiciones en las que la palabra de entrada no se corresponde con el patrón especificado en la regla.

Los transductores de estados finitos que representan las reglas ortográficas se diseñan de tal forma que cualquier cadena de morfemas que no cumpla las restricciones establecidas por ellas pasa a través del transductor sin sufrir ningún cambio. Esto permite que se puedan concatenar **diferentes transductores para aplicar distintas reglas ortográficas** a una misma cadena de caracteres.

De hecho, el conjunto de los transductores de estados finitos que representan cada una de las diferentes reglas que componen la ortografía de un idioma pueden operar en paralelo o se pueden ejecutar en serie como una larga cascada de transductores. Utilizar una configuración en paralelo o en serie es una decisión de diseño a tomar cuando se implementan estos sistemas para el análisis morfológico.

Una vez se ha definido el conjunto de transductores de estados finitos que representan las reglas ortográficas de un idioma, estos se pueden combinar con un lexicón y con el transductor de estados finitos presentado en el apartado anterior y que permite traducir una simple cadena de morfemas concatenados en su representación a nivel léxico. Entonces al introducir las reglas ortográficas en el sistema que analiza automáticamente la morfología de las palabras, se añade un segundo nivel al análisis morfológico que se conoce como **morfología de dos niveles** (Koskenniemi, 1983).

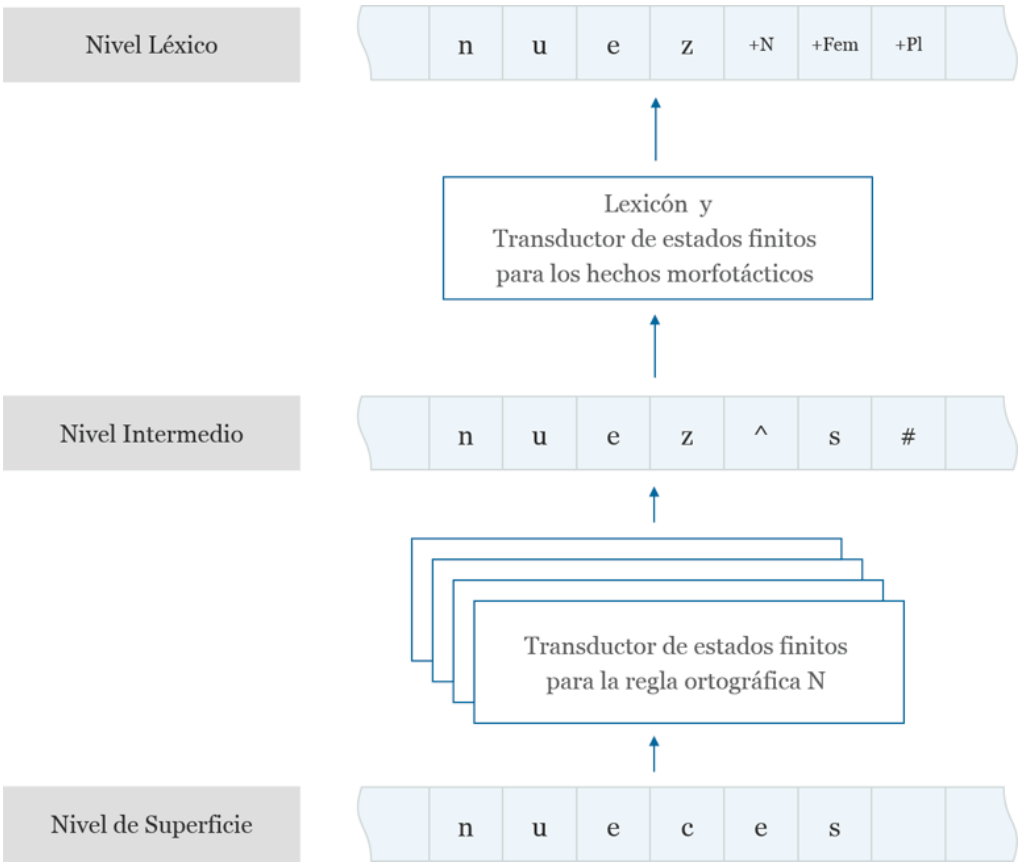


Figura 14. Análisis morfológico de dos niveles.

Análisis morfológico de dos niveles

La figura 7 muestra el funcionamiento del **análisis morfológico de dos niveles:**

- En un primer nivel:
  - La entrada a nivel de superficie es la cadena de caracteres que representa la palabra y es procesada por el conjunto de transductores de estados finitos que representan las reglas ortográficas del idioma.
  - Estos transductores son los encargados de detectar los posibles cambios ortográficos que hayan sufrido los morfemas al combinarse para crear la palabra.
  - Como salida de este primer nivel se obtiene la concatenación los diferentes morfemas que conforman la palabra, lo que se llama representación a **nivel intermedio**.

- En el segundo nivel opera el transductor de estados finitos que representa los hechos morfotácticos del lexicón, es decir, la información sobre cómo combinar o concatenar morfemas para generar las palabras.
  - Por lo que la representación a nivel intermedio, que es una simple concatenación de morfemas, es procesada por este transductor léxico y se determinan el lema o raíz de la palabra y las características morfológicas de la misma.
  - Al final se obtiene una representación a nivel léxico que recoge la información morfológica de la palabra.

Ejemplo ilustrativo 5.

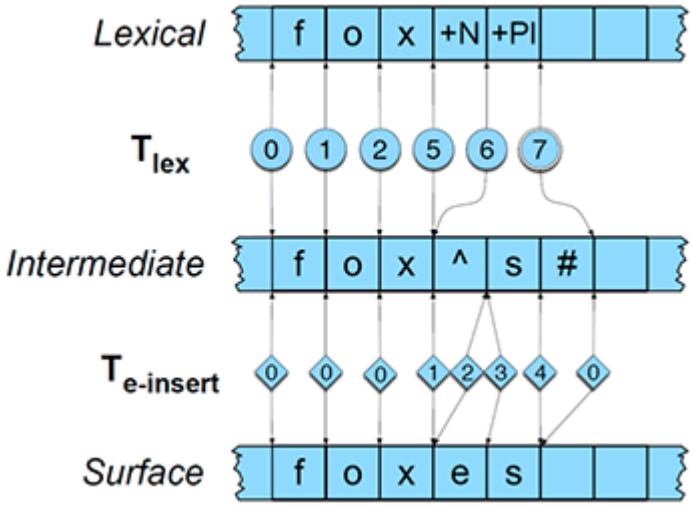


Figura 15. Análisis morfológico de la palabra inglesa *«foxes»*.
Fuente: Jurafsky y Martin, 2009.

Análisis morfológico de la palabra inglesa *«foxes»* (zorros) utilizando el transductor de estados finitos que recoge información morfotáctica de cómo generar los plurales (ejemplo ilustrativo 3, figura 5) y el transductor de estados finitos que modela la regla ortográfica para añadir la «e» al crear el plural (ejemplo ilustrativo 4, figura 6).

En esta figura 8 se observa la concatenación de dos transductores de estados finitos, $T_{e-insert}$ y T_{lex}, para realizar el análisis morfológico.

- Se llama $T_{e-insert}$ al transductor de estados finitos que **modela la regla ortográfica** para añadir la «e» al crear el plural de algunos nombres en inglés y que se ha presentado en la figura 6 del ejemplo ilustrativo 4.

- Se llama T_{lex} al transductor de estados finitos que **recoge información morfotáctica** de cómo generar los plurales en inglés y que se ha presentado en la figura 4 del ejemplo ilustrativo 3.

El transductor $T_{e-insert}$ se encarga de transformar la representación en el nivel de superficie, es decir, la palabra *«foxes»* en la representación de nivel intermedio «fox^s#». Esta representación intermedia, que recoge la concatenación de los morfemas *«fox»* y «-s», es luego transformada por el transductor T_{lex} en la representación a nivel léxico «fox+N+Pl». Esta última representación proporciona información sobre la palabra analizada *«foxes»*, indicando que su lema o raíz es *«fox»* y sus características morfológicas son nombre (+N) y de número plural (+Pl). Así, la salida del análisis morfológico es la representación a nivel léxico «fox+N+Pl».

Los números que aparecen dentro de los rombos situados junto a la indicación $T_{e-insert}$ indican los estados en los que se encuentra dicho transductor después de procesar cada uno de los símbolos del nivel de superficie y de generar el correspondiente símbolo en el nivel intermedio. Por ejemplo, al procesar la letra «x» en el nivel de superficie, el transductor $T_{e-insert}$ transita del estado q_0 al estado q_1 y genera como salida en el nivel intermedio el símbolo (x). Acto seguido y sin necesidad de procesar otro símbolo a la entrada, el transductor transita desde el estado q_1 al estado q_2 y genera a la salida de nivel intermedio el símbolo (^).

De forma similar, los números que aparecen dentro de los círculos situados junto a la indicación T_{lex} indican los estados en los que se encuentra dicho transductor después de procesar cada uno de los símbolos del nivel intermedio y de generar el correspondiente símbolo en el nivel léxico. Por ejemplo, al procesar la letra «x» en el nivel de intermedio el transductor T_{lex} transita del estado q_2 al estado q_5 y genera

como salida en el nivel léxico el símbolo (x). Seguidamente, también transita desde el estado q_5 al estado q_6 y genera a la salida de nivel léxico el símbolo (+N).

Los analizadores morfológicos de dos niveles pueden ser difíciles de gestionar si se componen de muchos transductores de estados finitos. Sin embargo, esto no se convierte en un problema porque un conjunto de transductores de estados finitos trabajando en serie o en paralelo **se pueden combinar en un único transductor**.

Combinación de diferentes transductores de estados finitos

A continuación, ilustramos un ejemplo de la combinación de diferentes transductores de estados finitos por intersección (∧) y composición (○) en la figura 9.

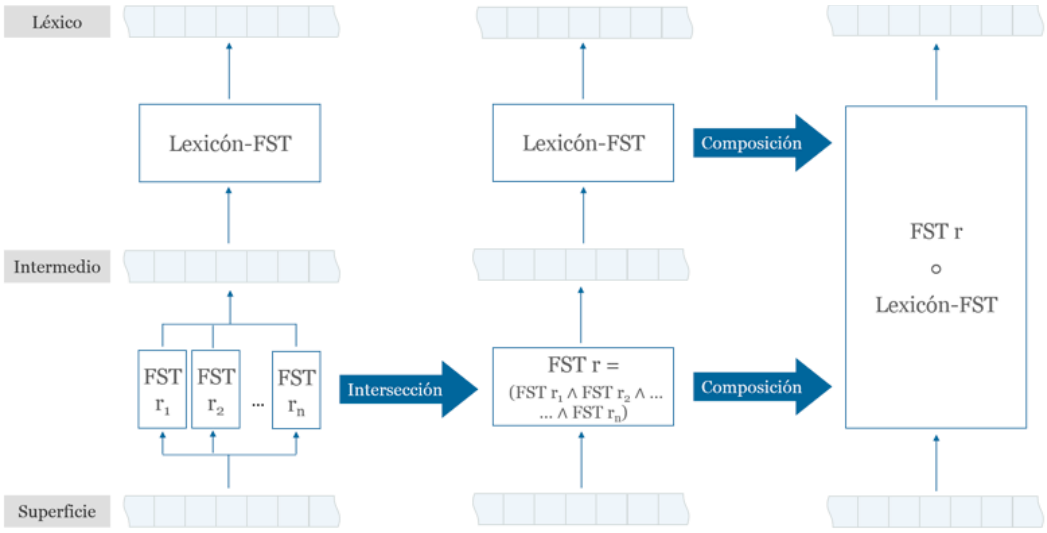


Figura 16. Arquitecturas alternativas para el análisis morfológico de dos niveles.

En paralelo

En el caso de un conjunto de transductores de estados finitos trabajando en paralelo como, por ejemplo, los transductores que modelan las diferentes reglas ortográficas $FST\ r_1, FST\ r_2, \ldots, FST\ r_n$ en la figura 9, se pueden combinar en un único transductor de estados finitos con una operación de **intersección** (∧).

$$FST\ r = (FST\ r_1 \wedge FST\ r_2 \wedge \ldots \wedge FST\ r_n)$$

El transductor resultante de la intersección $(FST\ r)$ en la imagen anterior simplemente aplica el producto cartesiano de los estados donde:

- Para cada estado q_i de un primer transductor y el estado q_j de un segundo transductor, se crea un nuevo estado q_{ij}.
- Entonces, para cualquier símbolo de entrada a, si el primer transductor pasa al estado q_n y el segundo transductor hace la transición al estado q_m, el transductor intersección pasa al estado q_{nm}.

En serie

En el caso de un conjunto de transductores de estados finitos trabajando en serie como, por ejemplo, el transductor resultante de modelar las reglas ortográficas que acabamos de ver $(FST\ r)$ y el transductor que representa los hechos morfotácticos del lexicón (Lexicón-FTS en la figura 9) se pueden combinar en un único transductor de estados finitos con una operación de **composición** (○).

$$FST\ r\ \circ\ Lexicón\text{-}FST$$

El transductor resultante de la composición genera para una secuencia S el mismo resultado que aplicar el primer transductor a la secuencia S y aplicar después el segundo transductor a la salida generada por el primero, lo que matemáticamente se representa como:

$$FST\ r\ \circ\ Lexicón\text{-}FST(S) = Lexicón\text{-}FST\big(FST\ r(S)\big)$$

Por lo tanto:

- Si la transición entre el estado q_i y el estado q_j del primer transductor viene marcada por el par de símbolos *a:b* y la transición entre los mismos estados para el segundo transductor se da para el par de símbolos *b:c*,
- Entonces la transición entre estos dos estados del transductor resultante de la composición vendrá marcada por el par de símbolos *a:c*.

Una de las grandes ventajas de utilizar transductores de estados finitos en el procesamiento del lenguaje natural es que los mismos transductores que realizan el análisis morfológico también son capaces de **generar lenguaje natural**: producir de forma automática las palabras a partir de la información sobre los morfemas que las componen. Esto va a ser muy útil a la hora de generar frases en los diálogos donde se debe encajar el género y el número de un nombre con el adjetivo que la acompaña o la persona del verbo con el sujeto de la frase.

Entonces, las arquitecturas para el análisis morfológico son útiles para generar lenguaje natural si se utilizan al revés, es decir, si toman como entrada la representación a nivel léxico y se genera como salida la palabra a nivel de superficie. Por ejemplo, si la arquitectura de dos niveles presentada en la figura 7 trabaja a la inversa y recibe la información léxica «nuez +N +Fem +Pl», va a ser capaz de generar la palabra «nueces».

Para **invertir los transductores** de estados finitos y que realicen esta función de generación del lenguaje natural, solo es necesario intercambiar las etiquetas de entrada y salida para cada una de las transiciones, así, el par de símbolos $s_1 : s_2$ (léxico:superficie) del transductor original se transforma en $s_2 : s_1$ en el transductor invertido.

Por último, la principal dificultad del análisis morfológico y que no se puede resolver solo con los transductores de estados finitos es la **ambigüedad**. Si a la entrada del sistema aparece la palabra «vino», el analizador morfológico no podrá decidir si esta representación a nivel de superficie se refiere a la representación léxica «venir +V +Perf +3P +Sg» o «vino +N +Masc +Sg». Para que el analizador morfológico pueda

decidir y no solo enumerar a la salida todas las posibles opciones, será necesario analizar algún tipo de información externa.

En este caso, si se conoce el **contexto** en el que aparece la palabra, qué otras palabras se encuentran a su alrededor, posiblemente se podrá determinar si la palabra es un nombre o un verbo. Se hablará de algunos algoritmos para tratar la ambigüedad y que hacen uso de información externa en temas posteriores.

Accede a los ejercicios de autoevaluación a través del aula virtual

2.7. Referencias bibliográficas

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics.* New Jersey, Estados Unidos: Prentice-Hall.

Koskenniemi, K. (1983). *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production* (Tesis doctoral, University of Helsinki, Finlandia). Recuperado de https://www.researchgate.net/publication/36084537_Two-level_Morphology_A_General_Computational_Model_for_Word-Form_Recognition_and_Production

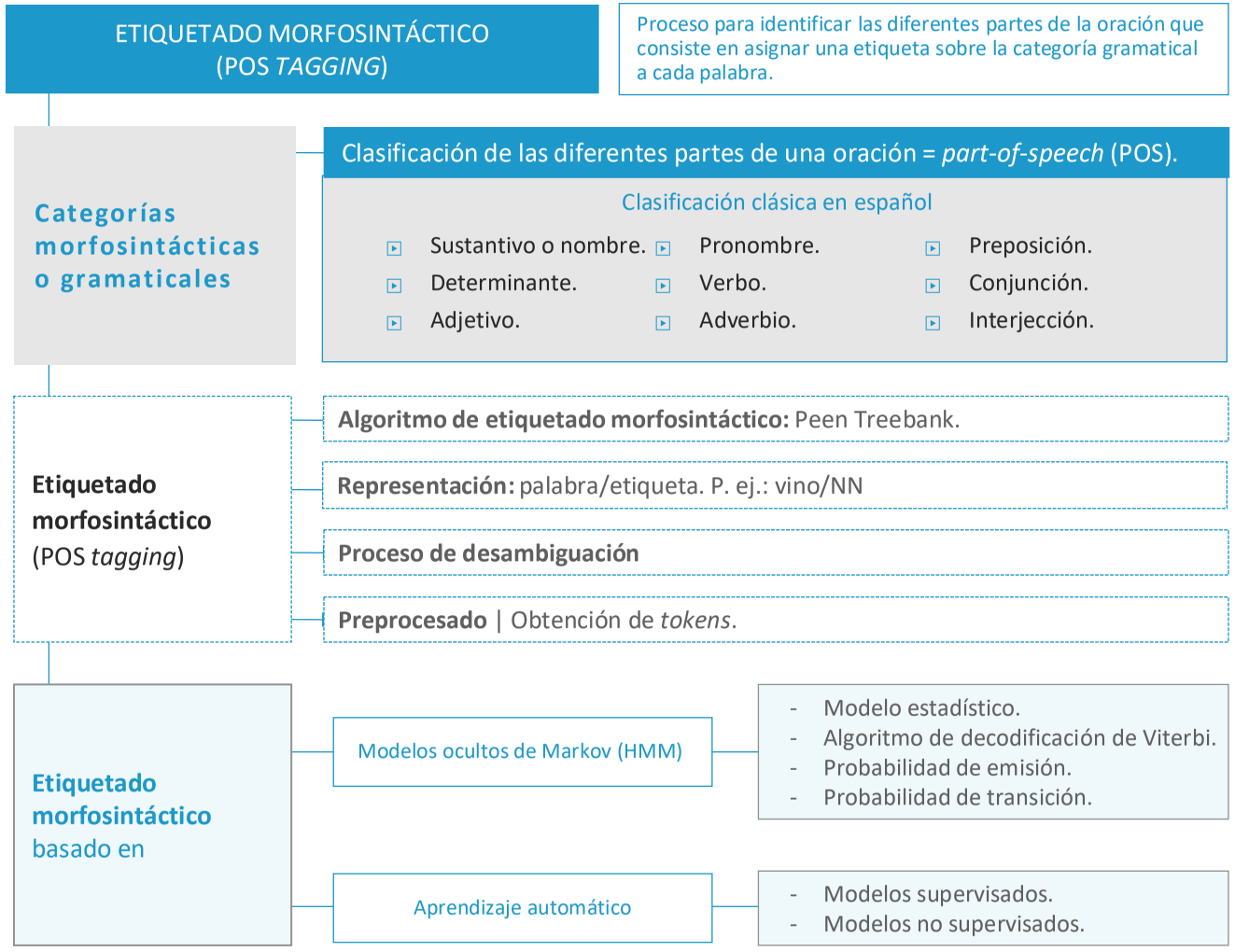
Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130-127.

Accede al vídeo «Resumen» a través del aula virtual

Tema 3. Etiquetado morfosintáctico (POS *tagging*)

Índice

Esquema



Ideas clave

3.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

En este tema se estudiará el etiquetado morfosintáctico y cómo calcularlo, haciendo especial hincapié en los modelos ocultos de Markov *(Hidden Markov Models)*. Los apartados de los que consta este tema son:

- Categorías morfosintácticas o gramaticales.
- Funcionamiento y características del etiquetado morfosintáctico.
- Etiquetado morfosintáctico basado en modelos ocultos de Márkov (HMM).
- Etiquetado morfosintáctico basado en aprendizaje automático.

Al finalizar el estudio de este tema serás capaz de:

- Identificar las diferentes categorías morfosintácticas o también llamadas gramaticales.
- Entender el funcionamiento y las características del etiquetado morfosintáctico.
- Aplicar un método estocástico basado en modelos ocultos de Markov (HMM) para realizar el etiquetado morfosintáctico.
- Describir diversos métodos basados en aprendizaje automático para realizar el etiquetado morfosintáctico.

Accede a los ejercicios de autoevaluación a través del aula virtual

3.2. Categorías morfosintácticas o gramaticales

Accede al vídeo «Categorías morfosintácticas» a través del aula virtual

La Real Academia Española (RAE, s. f.) en su diccionario de la lengua española define la palabra **morfosintaxis** como: «1. f. Ling. Parte de la gramática que integra la morfología y la sintaxis». Tal como se ha presentado en los temas anteriores, la morfología estudia la estructura de las palabras y la sintaxis, el modo en que se combinan las palabras. Por lo tanto, la morfosintaxis aúna la morfología y la sintaxis para determinar las diferentes **partes de la oración**, llamadas *part-of-speech* (POS) en inglés.

Las **categorías morfosintácticas** del lenguaje, que en español también se llaman categorías **gramaticales**, proporcionan una clasificación de las diferentes partes de la oración, es decir, una clasificación de las palabras según su tipo.

Las categorías gramaticales del español, según la clasificación clásica, son nueve: sustantivo o nombre, determinante, adjetivo, pronombre, verbo, adverbio, preposición, conjunción e interjección. Las definiciones de las diferentes clases de palabras o categorías gramaticales se presentan en la siguiente tabla.

Categorías morfosintácticas o gramaticales	
Sustantivo o nombre	Clase de palabras cuyos elementos poseen género y número, forman sintagmas nominales con diversas funciones sintácticas y designan entidades de diferente naturaleza.
Determinante	Clase de palabras cuyos elementos determinan al sustantivo o al grupo nominal y se sitúan generalmente en posición prenominal. El artículo definido y los demostrativos son determinantes.
Adjetivo	Clase de palabras cuyos elementos modifican a un sustantivo o se predican de él y denotan cualidades, propiedades y relaciones de diversa naturaleza. Ejemplos: inteligente, amplio, numérico, mismo, telefónico…

Pronombre	Clase de palabras cuyos elementos hacen las veces del sustantivo o del sintagma nominal y que se emplean para referirse a las personas, los animales o las cosas sin nombrarlos. P. ej.: ella, esto, quién…
Verbo	Clase de palabras cuyos elementos pueden tener variación de persona, número, tiempo, modo y aspecto.
Adverbio	Clase de palabras cuyos elementos son invariables y tónicos, están dotados generalmente de significado léxico y modifican el significado de varias categorías, principalmente de un verbo, de un adjetivo, de una oración o de una palabra de la misma clase.
Preposición	Clase de palabras invariables cuyos elementos se caracterizan por introducir un término, generalmente nominal u oracional, con el que forman grupo sintáctico.
Conjunción	Clase de palabras invariables, generalmente átonas, cuyos elementos manifiestan relaciones de coordinación o subordinación entre palabras, grupos sintácticos u oraciones.
Interjección	Clase de palabras invariables, con cuyos elementos se forman enunciados exclamativos, que manifiestan impresiones, verbalizan sentimientos o realizan actos de habla apelativos.

Tabla 5. Definiciones de las diferentes categorías morfosintácticas o categorías gramaticales según el diccionario de la lengua española de la Real Academia Española (RAE).

Conocer las partes de la oración (categorías morfosintácticas o gramaticales) es útil debido a la gran cantidad de información que brindan sobre una palabra y sus vecinos.

- Saber si una palabra es un sustantivo o un verbo nos dice mucho acerca de las **palabras vecinas**, por ejemplo, los sustantivos pueden ir precedidos de determinantes o seguidos de adjetivos.
- Y también sobre la **estructura sintáctica**, por ejemplo, los sustantivos son generalmente parte de los sintagmas nominales.

Por estas razones, el etiquetado morfosintáctico es un componente muy importante del análisis sintáctico que se va a estudiar en los siguientes apartados.

Accede a los ejercicios de autoevaluación a través del aula virtual

3.3. Funcionamiento y características del etiquetado morfosintáctico

Accede al vídeo «Categorías morfosintácticas» a través del aula virtual

El etiquetado morfosintáctico, llamado ***POS tagging*** *(part-of-speech tagging)* en inglés, es el proceso para identificar las diferentes partes de la oración y consiste en asignar una etiqueta (*tag*) sobre la categoría gramatical a cada una de las palabras de un texto de entrada.

La entrada del algoritmo de etiquetado morfosintáctico es una secuencia de palabras y la salida del algoritmo es una secuencia de pares formados por la palabra y la correspondiente etiqueta indicando la categoría gramatical a la que pertenece dicha palabra. Ante esto:

- Existen diferentes conjuntos de etiquetas que se pueden utilizar en el análisis morfosintáctico.
- Algunos etiquetadores morfosintácticos permiten indicar a su entrada el conjunto de etiquetas que:
  - Identifican cada categoría gramatical.
  - Se van a utilizar en el proceso de etiquetado de cada parte de la oración.

Hoy en día, la mayoría de los algoritmos de procesamiento del lenguaje natural que procesan palabras en inglés utilizan el **Penn Treebank** (Marcus, Santorini y Marcinkiewicz, 1993).

El Penn Treebank es un conjunto de 45 etiquetas que identifican las diferentes partes de la oración, tal como se muestra en la siguiente imagen.

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	*+,%, &*
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figura 17. Etiquetas para las categorías gramaticales en el Penn Treebank.
Fuente: Jurafsky y Martin, 2009.

En el etiquetado morfosintáctico se identifican las diferentes partes de la oración y se asigna una etiqueta a cada una de las palabras, tal y como se ha comentado anteriormente. La forma más habitual para **representar la salida del etiquetador** morfosintáctico es colocar, después de cada palabra, la etiqueta para la categoría gramatical separada por una barra.

Ejemplos. Si el etiquetador morfosintáctico analiza la frase «bebo un vaso del vino tinto», suponiendo que se utilizan las etiquetas para las categorías gramaticales definidas en el Penn Treebank, la salida sería:

```
bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

Para la frase «vino de un lugar lejano», el etiquetado morfosintáctico sería:

```
vino/VBZ de/IN un/DT lugar/NN lejano/JJ
```

En estos dos ejemplos de etiquetado morfosintáctico se observa que la misma palabra *vino* se etiqueta de forma distinta para las dos frases.

- En la primera frase, «bebo un vaso del vino tinto», la palabra *vino* pertenece a la categoría gramatical de los sustantivos o nombres (NN).
- Mientras que en la segunda frase, «vino de un lugar lejano», pertenece a la categoría gramatical de los verbos (VBZ).

Así, el etiquetado morfosintáctico realiza durante su funcionamiento un **proceso de desambiguación**: una palabra, que es ambigua y puede pertenecer a más de una categoría gramatical, se etiqueta correctamente según el contexto de la frase analizada.

Además, es importante notar que el algoritmo que realiza el etiquetado morfosintáctico ha separado la palabra *del* (que aparece en la frase «bebo un vaso del vino tinto») en las palabras *de* y *el* antes de etiquetarlas respectivamente como una preposición (IN) y un determinante (DT). Identificar que una palabra es una contracción y separarla en las dos que la constituyen forma parte del proceso previo de **preprocesado** de la oración, que permite separar la frase en las diferentes palabras.

La identificación de las palabras de una oración también es llamada proceso de **obtención de los *tokens***, porque *token* es el nombre inglés para definir una cadena de caracteres que representa una palabra y se realiza siempre previamente al etiquetado morfosintáctico y a otras tareas de procesamiento del lenguaje natural.

Accede a los ejercicios de autoevaluación a través del aula virtual

3.4. Etiquetado morfosintáctico basado en modelos ocultos de Markov (HMM)

Accede al vídeo «Construcción de un etiquetador morfosintáctico basado en modelos ocultos de Markov», «Probabilidades de transición y emisión», «Algoritmo de Viterbi» a través del aula virtual

Una de las técnicas más utilizadas en el etiquetado morfosintáctico es los modelos ocultos de Markov o HMM (por sus siglas del inglés, *Hidden Markov Model*). Esta técnica consiste en construir un modelo de lenguaje estadístico que se utiliza para obtener, a partir de una frase de entrada, la secuencia de etiquetas gramaticales que tiene mayor probabilidad.

Un modelo oculto de Markov es un **modelo estadístico** que se puede representar como una máquina de estados finitos, pero donde las transacciones entre estados son probabilísticas y no determinísticas. El objetivo es determinar los parámetros desconocidos (ocultos) a partir de los parámetros observables.

Por ejemplo, si hemos etiquetado una palabra como determinante, la próxima palabra será un nombre con un 40 % de probabilidad, un adjetivo con otro 40 % y un número el 20 % restante. Conociendo esta información, un sistema puede decidir que la palabra «vino» en la frase «el vino» es más probable que sea un nombre a que sea un verbo.

Para el etiquetado morfosintáctico, los HMM son entrenados en un conjunto de datos totalmente etiquetados, esto es, un conjunto de frases con cada palabra anotada con una etiqueta describiendo su categoría gramatical. A partir de los datos de entrenamiento, los HMM fijan estimaciones de máxima verosimilitud para cada uno de los estados y determina las diferentes probabilidades que rigen el modelo.

Para llevar a cabo el proceso de estimación de probabilidades se usa el **algoritmo de decodificación de Viterbi** (Forney, 1973). El objetivo de decodificación HMM es elegir la secuencia de etiquetas más probable dada la secuencia de observación de n palabras w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

Esta ecuación la podemos reescribir mediante el uso de la regla de Bayes de la siguiente forma:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

También podemos simplificar esta ecuación eliminando el denominador $P(w_1^n)$:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$$

Los etiquetadores HMM hacen dos **suposiciones** que permiten simplificar estas ecuaciones aún más:

1. La primera es que la probabilidad de aparición de una palabra depende solo de su propia etiqueta y es independiente de las palabras y etiquetas vecinas:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i^n | t_i^n)$$

2. La segunda suposición, también llamada bigrama o digrama, es que la probabilidad de una etiqueta solo depende de la etiqueta anterior, en lugar de toda la secuencia de etiquetas:

$$Pt_i^n \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Aplicando estas suposiciones a las ecuaciones anteriores terminamos con la siguiente ecuación para la secuencia de etiquetas más probable de un etiquetador bigrama, las cuales corresponden a la **probabilidad de emisión y la probabilidad de transición** de un HMM:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i^n|t_i^n)}^{\boldsymbol{emission}} \overbrace{P(t_i|t_{i-1})}^{\boldsymbol{transition}}$$

Ejemplos de probabilidad de transición y probabilidad de emisión

Veamos a través de un ejemplo cómo se calculan y se utilizan en una tarea de etiquetado estas probabilidades. En el etiquetado HMM, las probabilidades se estiman simplemente a partir de un corpus de entrenamiento etiquetado; para este ejemplo vamos a utilizar el **corpus etiquetado WSJ,** una colección de un millón de palabras que se publicaron en los artículos del Wall Street Journal (WSJ) en 1989 y que están anotadas utilizando las etiquetas morfosintácticas del Penn Treebank.

Un **corpus lingüístico** es una colección de textos representativos de una lengua que se utilizan para el análisis lingüístico. Los corpus pueden estar anotados o etiquetados de forma que las palabras que lo conforman presentan, además, algún tipo de información lingüística.

Accede al corpus a través del aula virtual o desde la siguiente dirección web:
https://catalog.ldc.upenn.edu/LDC2000T43

Probabilidades de transición

Las probabilidades de transición de etiqueta $P(t_i|t_{i-1})$ representan la probabilidad de una etiqueta dada la etiqueta anterior. Por ejemplo, los verbos modales (etiqueta MD) como *«can»* (poder) son muy probablemente seguidos por un verbo en la forma base (etiqueta VB) como *«run»* (correr), por lo que espera que esta probabilidad sea alta.

La estimación de **máxima verosimilitud** de una probabilidad de transición se calcula por recuento de las veces que vemos la primera etiqueta en un corpus etiquetado y la frecuencia con que esta primera etiqueta es seguida por la segunda según la siguiente fórmula:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

En el corpus WSJ, por ejemplo, los MD aparecen 13 124 veces, de las cuales son seguidos por un VB 10 471 veces, lo cual resulta en una estimación de máxima probabilidad de:

$$P(VB|MD) = \frac{C(MD, VB)}{C(MD)} = \frac{10471}{13124} = .80$$

Probabilidades de emisión

Las probabilidades de emisión $P(w_1^n|t_1^n)$ representan la probabilidad de que, dada una etiqueta (digamos MD), esta se asocie con una palabra concreta (digamos *«will»*). La estimación de **máxima verosimilitud** de la probabilidad de emisión en general se define como:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

De los 13 124 casos de MD en el corpus WSJ, estas se asocian o refieren a *«will»* en 4046 ocasiones, por tanto:

$$P(will|MD) = \frac{C(MD, will)}{C(MD)} = \frac{4046}{13124} = .31$$

Como aclaración, hemos de decir que esta probabilidad no se refiere a «cuál es la etiqueta más probable para la palabra *'will'*», ya que esta sería la probabilidad *a posteriori* $P(MD|will)$. En su lugar, $P(will|MD)$ responde a una pregunta ligeramente menos intuitiva, concretamente «si vamos a generar una etiqueta MD, ¿qué probabilidades hay de que este MD sea *'wil'*?».

Los dos tipos de probabilidades mencionados anteriormente, la probabilidad de transición $P(VB|MD)$ y la probabilidad de emisión $P(will|MD)$, se corresponden al conjunto A de probabilidades de transición del HMM y al conjunto B de probabilidades de observación *B* del HMM.

La figura 2 ilustra algunas de las probabilidades de transición A para tres estados en un etiquetador morfosintáctico HMM; el etiquetador completo tendría un estado para cada etiqueta. En esta imagen, las probabilidades de transición A se utilizan para calcular la probabilidad *a priori*.

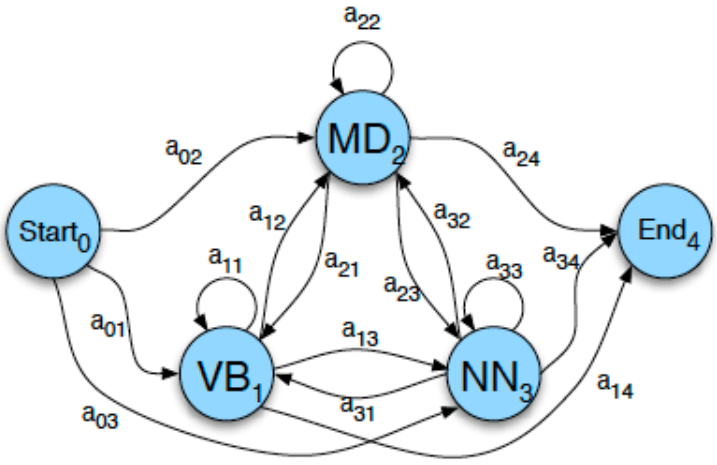


Figura 18. Un trozo de la cadena de Markov correspondientes a los estados ocultos del HMM.
Fuente: Jurafsky y Martin, 2009.

La figura 3 muestra otra vista de estos tres estados, pero centrándose en algunas de las probabilidades de observación B de cada palabra. Cada estado oculto está asociado con un vector de probabilidades para cada palabra en observación.

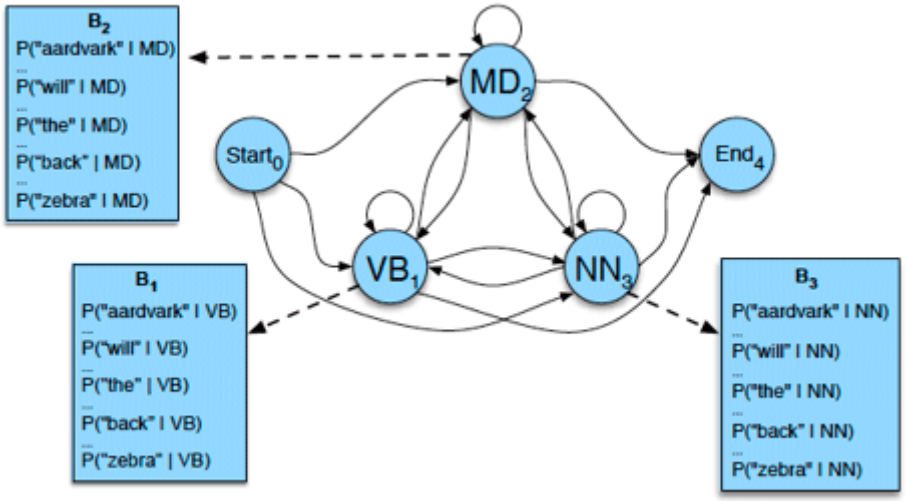


Figura 19. Algunas de las probabilidades de observación *B* para el HMM de la figura 2.
Fuente: Jurafsky y Martin, 2009.

Como indicábamos sobre esta última imagen, cada estado está asociado con un vector de probabilidades con una probabilidad para cada posible palabra en observación, a excepción de los estados no emisores de inicio y fin.

Ejemplo. Finalmente, vamos a trabajar a través de un ejemplo de cálculo de la mejor secuencia de etiquetas que corresponde a la siguiente secuencia de palabras: «*Janet will back the bill*» (en español, «Janet respaldará la ley»). La secuencia de etiquetas correcta es:

```
Janet/NNP will/MD back/VB the/DT bill/NN
```

Sea el modelo HMM el definido por el conjunto A de probabilidades de transición (figura 4), y el conjunto B de probabilidades de observación (figura 5). Cada elemento a_{ij} del conjunto A describe la probabilidad de transitar de un estado oculto i (etiqueta i) a otro estado oculto j (etiqueta j). Cada elemento $b_i(o_t)$ describe la probabilidad de observar las palabras dadas las etiquetas.

	PNN	MD	VB	JJ	NN	RB	DT
<S>	0.2767	0,0006	0,0031	0,0453	0,0449	0,0510	0.2026
PNN	0.3777	0,0110	0,0009	0,0084	0,0584	0,0090	0,0025
MD	0,0008	0,0002	0.7968	0,0005	0,0008	0.1698	0,0041
VB	0,0322	0,0005	0,0050	0,0837	0,0615	0,0514	0.2231
JJ	0,0366	0,0004	0,0001	0,0733	0.4509	0,0036	0,0036
NN	0,0096	0,0176	0,0014	0,0086	0.1216	0,0177	0,0068
RB	0,0068	0,0102	0.1011	0.1012	0,0120	0,0728	0,0479
DT	0.1147	0,0021	0,0002	0.2157	0.4744	0,0102	0,0017

Figura 20. Conjunto A de probabilidades de transición $P(t_i|t_{i-1})$ calculadas a partir del corpus WSJ. Fuente: Jurafsky y Martin, 2009.

En la imagen anterior vemos que cada fila representa el evento condicionante; por ejemplo:

$$P(VB|MD) = 0.7968.$$

La siguiente imagen (figura 5) se obtiene a partir del recuento de apariciones de una palabra en el corpus. Así:

- La palabra *Janet* solo aparece como un nombre propio (NNP).
- La palabra *will* aparece en el corpus como tres categorías gramaticales diferentes; puede ser:
  - Un verbo modal (MD) para generar el futuro de los verbos en inglés.
  - Un verbo (VB) que significa «desear» en español.
  - O un nombre (NN) que significa «deseo» o «voluntad» en español.

	Janet	will	back	the	bill
PNN	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figura 21. Conjunto B de probabilidades de observación calculadas a partir del corpus WSJ. Fuente: Jurafsky y Martin, 2009.

La figura 6 muestra un esquema con las posibles etiquetas para el ejemplo anterior, así como la correcta secuencia de etiquetado final. Esta secuencia del etiquetado morfosintáctico se ha calculado aplicando **el algoritmo de Viterbi**, cuyo pseudocódigo se presenta en la figura 7.

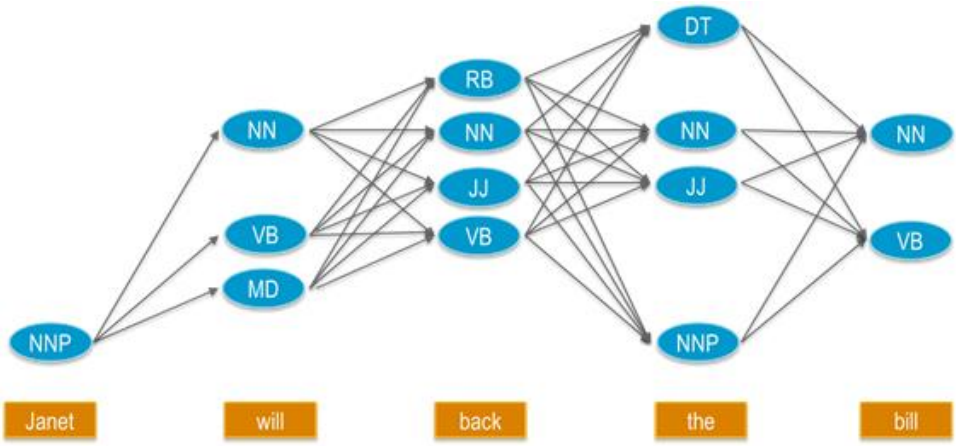


Figura 22. Diagrama de la tarea de etiquetado para la frase ejemplo.

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ; initialization step
  viterbi$[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  backpointer$[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    viterbi$[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    backpointer$[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
viterbi$[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
backpointer$[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from *backpointer*$[q_F,T]$

Figura 23. Pseudocódigo del algoritmo de Viterbi para encontrar la secuencia óptima de *tags* en un etiquetador morfosintáctico basado en HMM.
Fuente: Jurafsky y Martin, 2009.

El algoritmo de Viterbi crea una matriz de probabilidades con una columna para cada observación t y una fila para cada estado q_i de la máquina de estados finitos (o autómata finito) que representa el HMM. Para el ejemplo, el algoritmo crea N = 5

columnas de estado, la primera para la observación de la primera palabra «Janet», la segunda para «will» y así sucesivamente hasta completar las cinco palabras que conforman la frase, tal como se muestra en la figura 8.

Se empieza en la primera columna para establecer el valor de Viterbi en cada celda $v_t\ (i)$, que se calcula como el producto de la probabilidad de transición (hasta ese estado desde el estado inicial) por la probabilidad de observación (de la primera palabra). Entonces, se avanza columna a columna y, para cada estado en la columna 1, se calcula la probabilidad de transición a cada estado en la columna 2, y así sucesivamente.

Para cada estado q_i en el tiempo t, se calcula valor de Viterbi (representado como `viterbi[s,t]` en el pseudocódigo de la figura 7) tomando el máximo sobre las extensiones de todas las rutas que conducen a la celda actual, según la siguiente ecuación:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

Donde:

- $v_{t-1}(i)$ es la probabilidad de la ruta de Viterbi previa, es decir, la ruta para el tiempo anterior o en la observación $t-1$.
- a_{ij} es la probabilidad de transición del estado anterior q_i al estado actual q_j.
- $b_j(o_t)$ es la probabilidad de observación del símbolo o_t dado el estado actual q_j.

Entonces, cada celda de la primera columna donde aparece la palabra «Janet» se calcula multiplicando la probabilidad de Viterbi previa en el estado de inicio q_0, que es $v_0\ (0) = \ 1.0$ por la probabilidad de transición desde el estado de inicio q_0 hasta la etiqueta para esa celda, por ejemplo:

$P(NNP|start) \ = \ 0.2767$ para la celda en la que la etiqueta es NNP.

Y por la probabilidad de observación de la palabra «Janet» dada la etiqueta de esa celda, por ejemplo:

$P(Janet|NNP) = 0.000032$ para la celda en la que la etiqueta es NNP.

Por lo tanto:

- $v_1\,(1) = 1.0 \cdot 0.2767 \cdot 0.000032 = 0.000009$ para la celda en la que la etiqueta es NNP para la columna donde la palabra es «Janet».
- El resto de las celdas en esta columna son cero, ya que la palabra «Janet» no puede tener asociada ninguna de las otras etiquetas gramaticales.

A continuación, cada celda en la columna *«will»* se actualiza con la ruta de probabilidad máxima desde la columna anterior. En la figura 8 se muestran los valores para las celdas MD, VB y NN. Cada celda obtiene los siete valores de la columna anterior multiplicados por la probabilidad de transición apropiada; se toma el valor máximo, $v_1\,(1) \cdot P(MD|NNP)$, que proviene del estado NNP en la columna anterior y este valor se multiplica por la probabilidad de observación del símbolo *«will»* dada la etiqueta correspondiente a la celda en cuestión. Por ejemplo, para la celda MD el valor de Viterbi es:

$$v_2\,(2) = v_1\,(1) \cdot P(MD|NNP) \cdot P(will|MD) = 0.00000002772$$

Se continúa aplicando el algoritmo columna a columna hasta llegar al final.

Para la probabilidad máxima de Viterbi, se traza la inversa para obtener la ruta concreta que ha llevado a ese valor y que se corresponde con la mejor secuencia de etiquetas: NNP MD VB DT NN.

Entonces, esta secuencia de etiquetas se corresponde con el etiquetado morfosintáctico de la frase:

```
Janet/NNP will/MD back/VB the/DT bill/NN
```

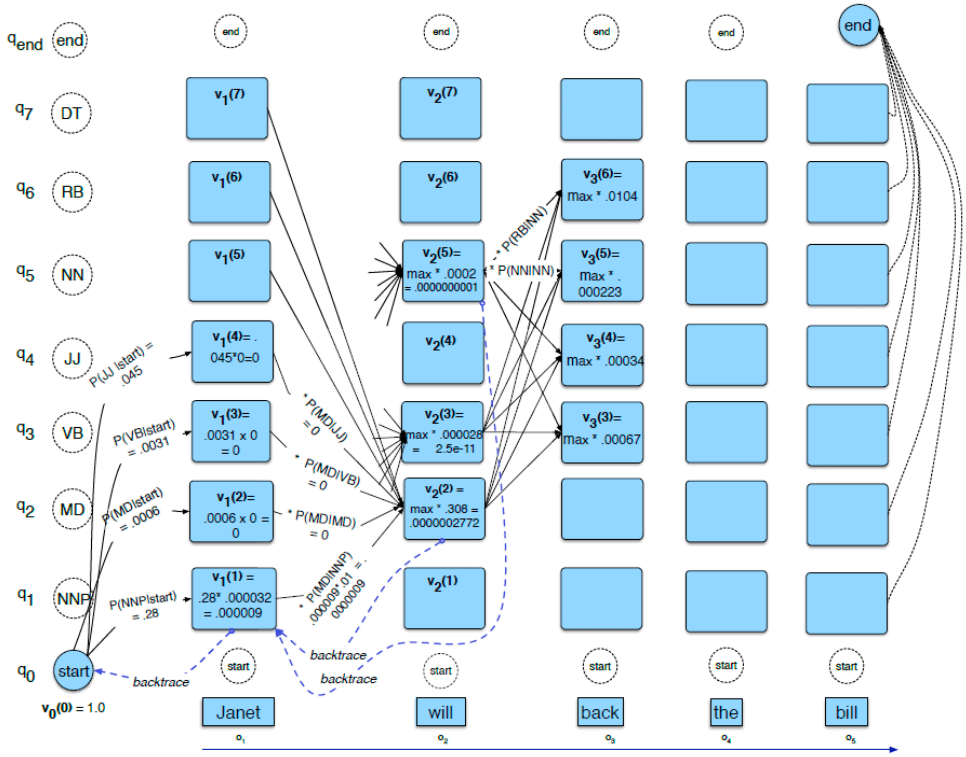


Figura 24. Matriz para la aplicación del algoritmo de Viterbi en el etiquetado morfosintáctico de la frase «*Janet will back the bill*». Fuente: Jurafsky y Martin, 2009.

Accede a los ejercicios de autoevaluación a través del aula virtual

3.5. Etiquetado morfosintáctico basado en aprendizaje automático

Accede al vídeo «Construcción de un etiquetador morfosintáctico basado en modelos ocultos de Markov», «Probabilidades de transición y emisión», «Algoritmo de Viterbi» a través del aula virtual

Los modelos más vanguardistas de etiquetado morfosintáctico utilizan diversas técnicas de aprendizaje automático tanto supervisado como no supervisado. Ejemplos de modelos supervisados son, entre otros:

- El algoritmo perceptrón (Collins, 2002).
- El modelo logaritmo lineal bidireccional (Toutanova, Klein, Manning y Singer, 2003).
- Las máquinas de vectores soporte (Giménez y Márquez, 2004).

No obstante, tanto estos como los algoritmos presentados en el tema dependen en gran medida del dominio de datos de entrenamiento, así como el etiquetado marcado por los expertos. Algunos trabajos recientes en etiquetado morfosintáctico se centran en buscar alternativas que permitan relajar estas condiciones; es el caso de los algoritmos no supervisados que etiquetan clústers de palabras en clases morfosintácticas (Christodoulopoulos, Goldwater, y Steedman, 2010) (Sirts, Eisenstein, Elsner, y Goldwater, 2014).

Muchos algoritmos se basan en la combinación de datos etiquetados con datos no etiquetados, por ejemplo, usando **coentrenamiento** (Søgaard, 2010). La asignación de etiquetas a texto de muy distintos tipos como, por ejemplo, aquellos provenientes de Twitter, puede requerir la adición de nuevas etiquetas para las direcciones URL (URL), nombre de usuario menciona (USR), *retweets* (RT), y *hashtags* (HT); la normalización de las palabras no estándar y técnicas *bootstrapping* para emplear datos no supervisados (Derczynski et al., 2013).

Accede a los ejercicios de autoevaluación a través del aula virtual

3.6. Referencias bibliográficas

Christodoulopoulos, C., Goldwater, S. y Steedman, M. (2010). Two decades of unsupervised POS induction: How far have we come? En *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 575-584). Massachusetts, Estados Unidos: Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. En *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1-8). Filadelfia, Estados Unidos: Association for Computational Linguistics.

Derczynski, L., Ritter, A., Clark, S. y Bontcheva, K. (2013). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. En *Proceedings of Recent Advances in Natural Language Processing* (pp. 198-206). Hissar, Bulgaria: Association for Computational Linguistics.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE, 61*(3), 268-278.

Giménez, J. y Márquez, L. (2004). SVMTool: A general POS tagger generator based on Support Vector Machines. En *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Lisboa, Portugal: LREC.

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics.* New Jersey (Estados Unidos): Prentice-Hall.

Marcus, M. P., Santorini, B. y Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics, 19*(2), 313-330.

RAE. (S. f.). Morfosintaxis. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=PpC0akB

Sirts, K., Eisenstein, J., Elsner, M., y Goldwater, S. (2014). POS induction with distributional and morphological information using a distance-dependent Chinese restaurant process. En *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (pp. 265-271). Baltimore, Estados Unidos. Recuperado de http://www.aclweb.org/anthology/P14-2044

Søgaard, A. (2010). Simple semi-supervised training of part-of-speech taggers. En *Proceedings of the ACL 2010 Conference Short Papers* (pp. 205-208). Uppsala, Suecia: Association for Computational Linguistics.

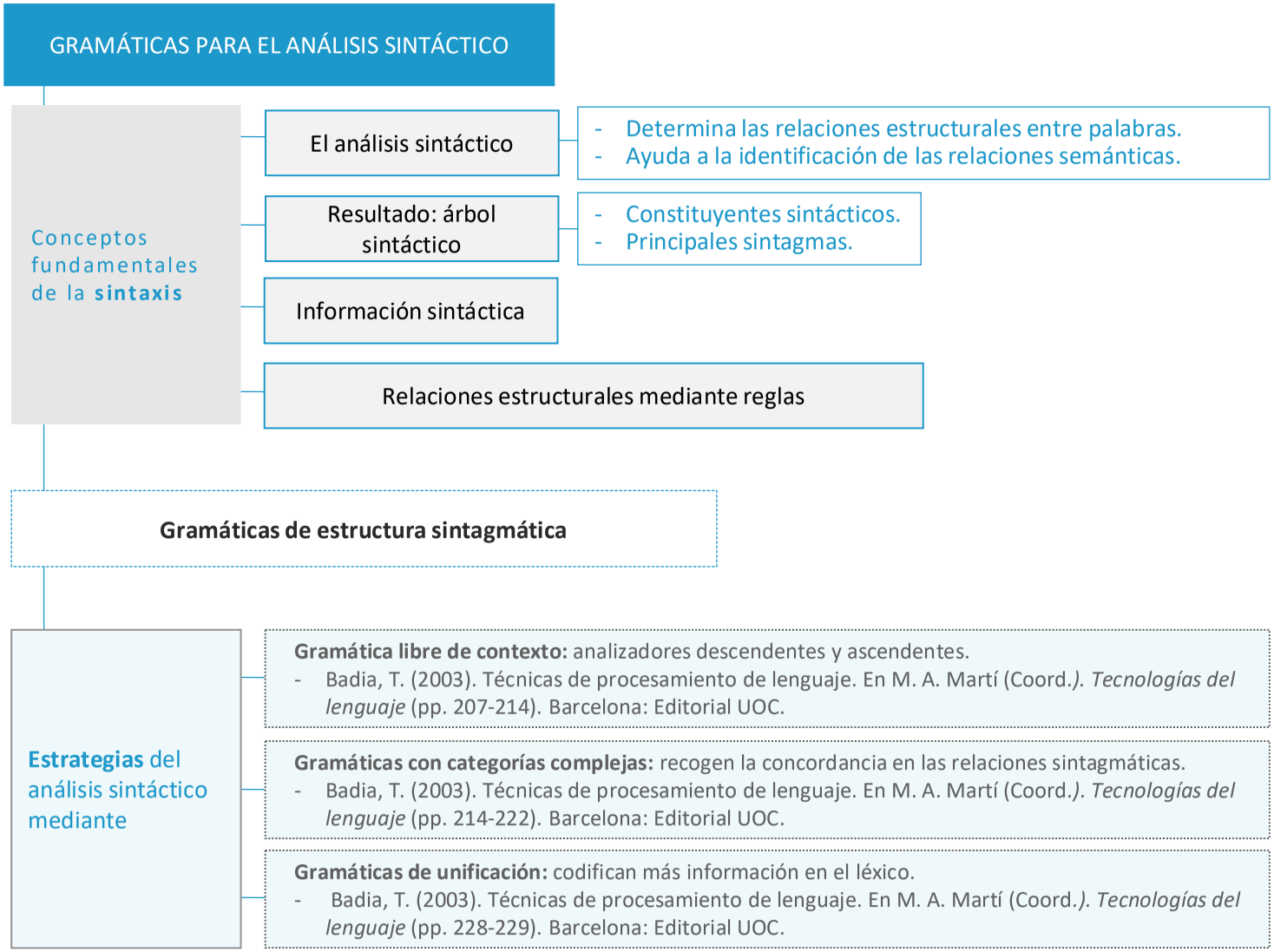
Toutanova, K., Klein, D., Manning, C. D. y Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. En *Proceedings of HLT-NAACL* (pp. 173-180). Edmonton, Canadá: Association for Computational Linguistics.

Accede al vídeo «Resumen» a través del aula virtual

Tema 4. Gramáticas para el análisis sintáctico

Índice

Esquema



Ideas clave

4.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

En este tema profundizaremos en el análisis sintáctico y las relaciones estructurales entre las palabras. Esta información sintáctica se modela mediante las gramáticas sintagmáticas, de las cuales veremos las gramáticas de estados finitos, las libres de contexto, las sensibles al contexto, las gramáticas recursivamente enumerables y las gramáticas generativo-transformacionales.

Los apartados de los que consta este tema son:

- Sintaxis.
- Gramáticas de estructura sintagmática.
- Estrategias del análisis sintáctico utilizando una gramática libre de contexto.
- Gramáticas con categorías complejas (o gramáticas de unificación).
- Análisis sintáctico con gramáticas de unificación.
- Gramáticas de dependencias (o gramáticas valenciales).

Al finalizar el estudio de este tema serás capaz de:

- Describir los conceptos fundamentales de la sintaxis.
- Definir el concepto y elementos de una gramática de estructura sintagmática que permite modelar la estructura formal del lenguaje.
- Identificar los diferentes tipos de gramáticas sintagmáticas.
- Entender el funcionamiento de las diferentes estrategias del análisis sintáctico utilizando una gramática libre de contexto: analizadores descendentes y analizadores ascendentes.

- Definir el concepto de gramática con categorías complejas y justificar su uso para recoger la concordancia en las relaciones sintagmáticas.
- Entender el funcionamiento de los analizadores sintácticos que utilizan una gramática de unificación y codifican más información en el léxico que en la gramática.
- Definir el concepto de gramática de dependencias o gramática valencial e identificar sus diferencias respecto a las gramáticas de estructura sintagmática.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.2. Sintaxis

Accede al vídeo «Conceptos fundamentales de sintaxis» a través del aula virtual

La Real Academia Española (RAE, s. f.) en su diccionario de la lengua española define la palabra **sintaxis:** «Del lat. tardío *syntaxis*, y este del gr. *σύνταξις sýntaxis*, de *συντάσσειν syntássein* "disponer conjuntamente", "ordenar". 1. f. Gram. Parte de la gramática que estudia el modo en que se combinan las palabras y los grupos que estas forman para expresar significados, así como las relaciones que se establecen entre todas esas unidades».

El análisis sintáctico

En el análisis sintáctico se determinan las relaciones estructurales entre palabras y es muy relevante en el procesamiento del lenguaje natural no solo por la información sintáctica que aporta, también porque es un paso esencial para la posterior identificación de las relaciones semánticas de las oraciones.

Árbol sintáctico. Es el resultado del análisis sintáctico. Sus nodos son los constituyentes sintácticos y las hojas, las palabras que componen la oración analizada. La estructura jerárquica del árbol permite observar que un constituyente está formado por una o varias palabras y por otros constituyentes.

Un **constituyente sintáctico** es una palabra o una secuencia de palabras que realizan una función conjunta dentro de la estructura jerárquica de la oración.

En la gramática tradicional se llama sintagma al constituyente sintáctico compuesto de dos a más elementos y según la *Nueva gramática de la lengua española* (Real Academia Española, 2009) se denomina grupo sintáctico.

Los principales tipos de sintagma de la lengua española se presentan en la siguiente tabla:

Sintagma	Núcleo	Función
Sintagma Nominal (SN)	Sustantivo Pronombre	Sujeto. Atributo. Complemento directo. Complemento indirecto (solo si el núcleo es un pronombre). Complemento predicativo. Complemento circunstancial.
Sintagma Verbal (SV)	Verbo	Predicado (predicado verbal o predicado nominal).
Sintagma Preposicional (SP o SPrep)	Preposición	Complemento de régimen. Complemento directo. Complemento preposicional. Complemento circunstancial.
Sintagma Adjetival (SAdj)	Adjetivo	Complemento adyacente. Complemento predicativo.
Sintagma Adverbial (SAdv)	Adverbio	Adjunto. Complemento circunstancial.

Tabla 6. Principales sintagmas de la lengua española.

Información sintáctica. Es información relativa a las relaciones estructurales entre palabras. Dicha información de una palabra permite determinar las combinaciones posibles de este elemento con el resto de los elementos de la oración. Por ejemplo, la información sintáctica de un verbo puede indicar si este puede tener asociado un objeto directo, un objeto indirecto o un complemento preposicional. En concreto, los verbos transitivos exigen la presencia de un objeto directo y este conocimiento lingüístico sobre la estructura sintáctica de los verbos transitivos debe modelarse correctamente para poder realizar el análisis sintáctico.

Las relaciones estructurales entre palabras se pueden formular mediante reglas

Para indicar que en español un elemento que sea un sintagma nominal, por ejemplo, está compuesto por dos elementos, un determinante y un nombre (colocados en este orden específico) se puede utilizar la siguiente regla:

$$SN \rightarrow Det\ N$$

Donde:

- SN indica el sintagma nominal.
- Det: el determinante.
- N: el nombre.

A partir de esta información sintáctica se podrá determinar que el elemento «la mesa» es un sintagma nominal correcto porque se compone del determinante «la» seguido del nombre «mesa».

Una definición formal de la estructura sintáctica de una lengua es imprescindible para poder realizar correctamente el análisis sintáctico. Las **gramáticas de estructura sintagmática** que se presentan a continuación modelan la información sintáctica y, por tanto, recogen la información relativa a las relaciones estructurales de una lengua necesaria en el análisis sintáctico realizado en las tareas de procesamiento del lenguaje natural.

Cabe destacar que, en la lingüística moderna, se están extendiendo las llamadas **gramáticas valenciales** o **gramáticas de dependencias**. Estas gramáticas, a diferencia de las de estructura sintagmática, se basan en dependencias y no en los constituyentes sintácticos. Por lo que en las gramáticas de dependencias se modelan los elementos léxicos y las relaciones existentes entre estos elementos, tal como se muestra al final de este tema.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.3. Gramáticas de estructura sintagmática

Accede al vídeo «Estrategias descendentes de análisis sintáctico» a través del aula virtual

Para completar el estudio de esta sección deberás leer las **páginas 199-206** del siguiente libro: Badia, T. (2003). Técnicas de procesamiento de lenguaje. En M. A. Martí (Coord.). *Tecnologías del lenguaje* (pp. 199-206). Barcelona: Editorial UOC.
Disponible en la Biblioteca Virtual de UNIR.

Una gramática de estructura sintagmática permite definir la estructura formal del lenguaje. En esta sección se define el concepto y los elementos de una gramática de estructura sintagmática y, además, se identifican los diferentes tipos: gramáticas de estados finitos, gramáticas libres de contexto, gramáticas sensibles al contexto, gramáticas enumerables recursivamente y gramáticas generativas transformacionales.

Las páginas señaladas corresponden al apartado «1.2. Las gramáticas formales» del capítulo indicado para el estudio de esta sección.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.4. Estrategias del análisis sintáctico utilizando una gramática libre de contexto

Accede al vídeo «Estrategias ascendentes de análisis sintáctico» a través del aula virtual

Para completar el estudio de esta sección deberás leer las **páginas 207-214** del siguiente libro: Badia, T. (2003). Técnicas de procesamiento de lenguaje. En M. A. Martí (Coord.). *Tecnologías del lenguaje* (pp. 207-214). Barcelona: Editorial UOC.
Disponible en la Biblioteca Virtual de UNIR.

Las estratégicas más comúnmente aplicadas para el análisis sintáctico están basadas en el uso de gramáticas libres de contexto y dan lugar a dos tipos de **analizadores sintácticos**:

- Los descendentes.
- Los ascendentes.

Las páginas señaladas corresponden al apartado «1.3. Los analizadores» del capítulo indicado para el estudio de esta sección.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.5. Gramáticas con categorías complejas (o gramáticas de unificación)

Accede al vídeo «Estrategias ascendentes de análisis sintáctico» a través del aula virtual

Para completar el estudio de esta sección deberás leer las **páginas 214-222** del siguiente libro: Badia, T. (2003). Técnicas de procesamiento de lenguaje. En M. A. Martí (Coord.). *Tecnologías del lenguaje* (pp. 214-222). Barcelona: Editorial UOC.
Disponible en la Biblioteca Virtual de UNIR.

Una **gramática con categorías complejas**, también llamada gramática de unificación, permite aumentar la expresividad de la gramática y se utiliza en el caso en que sea necesario recoger la concordancia en las relaciones sintagmáticas, por ejemplo, que en sintagma nominal el nombre femenino singular debe ir acompañado por un determinante también femenino singular.

Las páginas señaladas corresponden al apartado «1.4. Las gramáticas con categorías complejas» del capítulo indicado para el estudio de esta sección.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.6. Análisis sintáctico con gramáticas de unificación

Accede al vídeo «Otras formas de gramáticas» a través del aula virtual

Para completar el estudio de esta sección deberás leer las **páginas 228-229** del siguiente libro: Badia, T. (2003). Técnicas de procesamiento de lenguaje. En M. A. Martí (Coord.). *Tecnologías del lenguaje* (pp. 228-229). Barcelona: Editorial UOC.
Disponible en la Biblioteca Virtual de UNIR.

Las gramáticas de unificación que resultan de adoptar categorías complejas permiten codificar más información en el léxico que en la gramática y así los **analizadores sintácticos** que utilizan estas **gramáticas de unificación** mejoran el proceso de análisis sintáctico.

Las páginas señaladas corresponden solo al punto 1 (gramáticas de unificación) del apartado «1.6. El análisis sintáctico» del capítulo indicado para el estudio de esta sección.

Accede a los ejercicios de autoevaluación a través del aula virtual

4.7. Gramáticas de dependencias o gramáticas valenciales

Accede al vídeo «Otras formas de gramáticas» a través del aula virtual

Una **gramática valencial** modela las dependencias entre los elementos léxicos de una oración y, de este hecho, proviene su nombre de **gramática de dependencias**. A diferencia de las gramáticas de estructura sintagmática, que modelan los constituyentes sintácticos de una oración, las gramáticas de dependencias describen la estructura sintáctica de una oración únicamente en términos de las palabras (o lemas) que la componen y del conjunto de las relaciones gramaticales establecidas entre las palabras.

Ejemplo ilustrativo 1. Análisis sintáctico utilizando una gramática de dependencias.

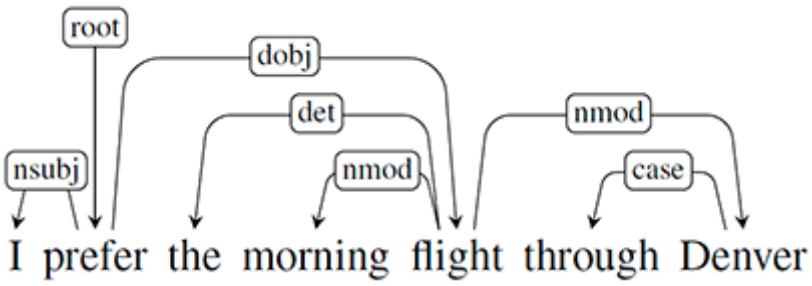


Figura 25. Resultado del análisis sintáctico utilizando una gramática de dependencias.
Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico de la oración en inglés *«I prefer the morning flight through Denver»* (en español, «Prefiero un vuelo a través de Denver por la mañana»), utilizando una gramática de dependencias, produciría la salida presentada en la figura 1. De la que se desprende que la palabra *«prefer»* (prefiero) es la raíz de la oración y que esta se relaciona a través de una dependencia del tipo *dobj* (objeto directo) con la palabra *«flight»* (vuelo). A su vez, la palabra *«flight»* se relaciona a través de una dependencia del tipo *det* (determinante) con la palabra *«flight»*.

El resultado del análisis sintáctico utilizando una gramática de dependencias (figura 1) se puede comparar con el árbol sintáctico (figura 2) resultante de realizar el análisis sintáctico de la misma oración utilizando una gramática de estructura sintagmática como podría ser una gramática libre de contexto.

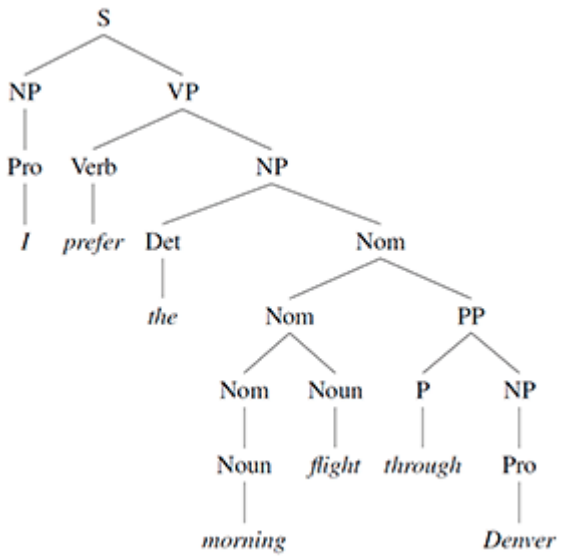


Figura 26. Árbol sintáctico que representan el resultado del análisis sintáctico utilizando una gramática de estructura sintagmática.
Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico utilizando gramáticas de dependencias se popularizó en la década de los 90. Por lo tanto, en las últimas décadas han aparecido diferentes analizadores sintácticos de dependencias basados en principios diversos: analizadores basados en programación dinámica (Eisner, 1996), en enfoques transicionales deterministas (Covington, 2001) (Nivre y Scholz, 2004), en aprendizaje automático supervisado (Yamada y Matsumoto, 2003) o en representaciones gráficas (McDonald et al., 2005).

Accede a los ejercicios de autoevaluación a través del aula virtual

4.8. Referencias bibliográficas

Covington, M. (2001). A fundamental algorithm for dependency parsing. En J. A. Miller y J. W. Smith (Eds.). *Proceedings of the 39th Annual ACM Southeast Conference* (pp. 95-102). Athens, Estados Unidos: Association for Computing Machinery.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: an exploration. En *Proceedings of the 16th International Conference on Computational Linguistics* (COLING-96) (pp. 340-345). Copenague, Dinamarca: [Association for Computational Linguistics]. Recuperado de http://aclweb.org/anthology/C96-1058

McDonald, R., Crammer, K. y Pereira, F. (2005). Online large-margin training of dependency parsers. En *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 91-98). Stroudsburg, Estados Unidos: Association for Computational Linguistics.

Nivre, J. y Scholz, M. (2004). Deterministic dependency parsing of English text. En *Proceedings of the 20th international conference on Computational Linguistics* p. 64-70). Stroudsburg, Estados Unidos: Association for Computational Linguistics.

RAE. (S. f.). Sintaxis. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=XzfiT9q

Real Academia Española. (2009). *Nueva gramática de la lengua española.* Barcelona: Espasa.

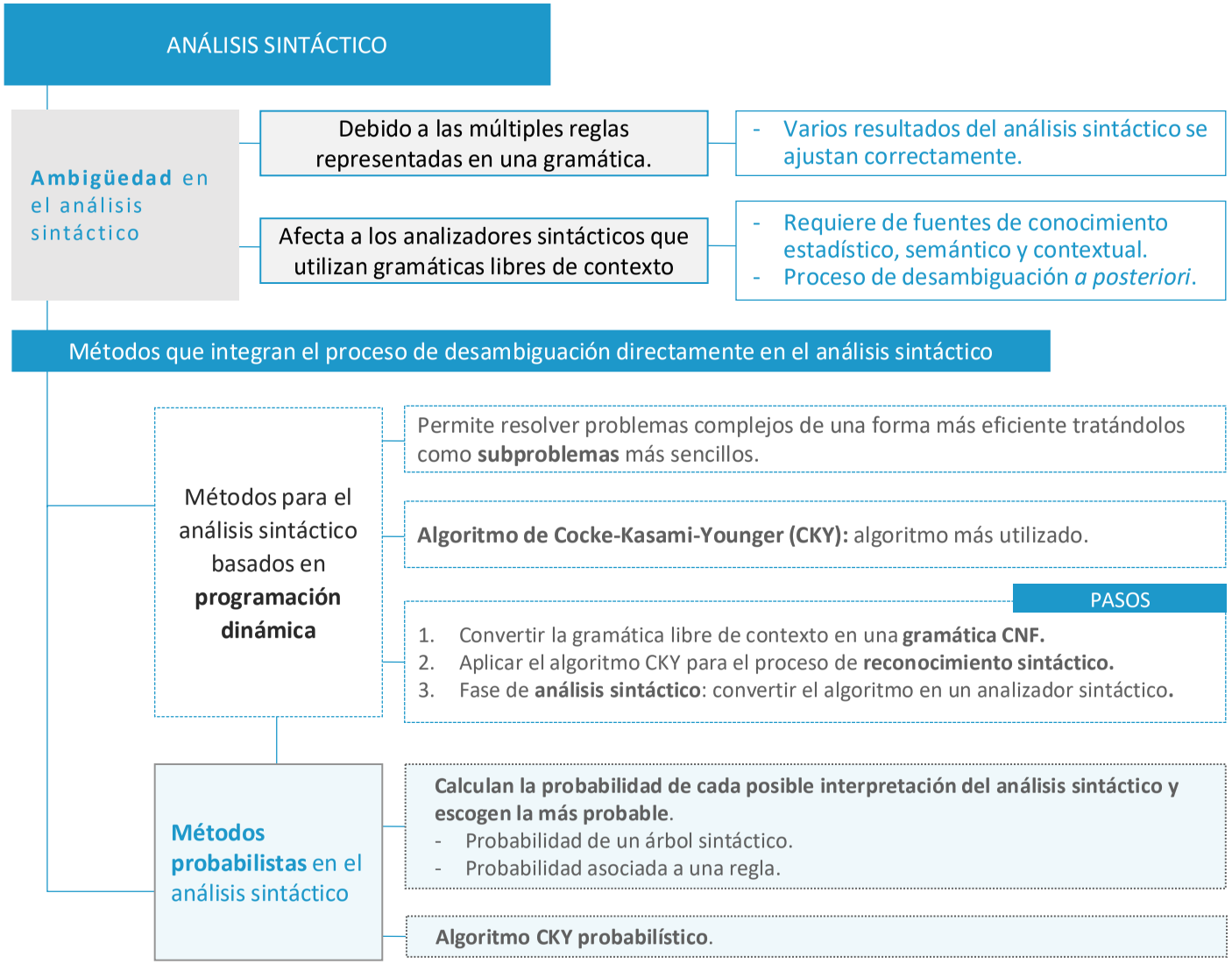
Yamada, H. y Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. En *The 8th International Workshop of Parsing Technologies (IWPT2003)* (pp. 195-206).

Accede al vídeo «Resumen» a través del aula virtual

Tema 5. Análisis sintáctico

Índice

Esquema



Ideas clave

5.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

En este tema profundizaremos en la ambigüedad estructural, uno de los mayores problemas que se dan en el análisis sintáctico, y cuáles son las técnicas para solucionarla como son los métodos basados en programación dinámica y los métodos probabilistas.

Los apartados de los que consta este tema son:

- Ambigüedad en el análisis sintáctico.
- Métodos para el análisis sintáctico basados en programación dinámica.
- Métodos probabilistas en el análisis sintáctico.
- Referencias bibliográficas.

Al finalizar el estudio de este tema serás capaz de:

- Entender cómo el problema de la ambigüedad en la estructura de las frases afecta al funcionamiento de los analizadores sintácticos basados en búsqueda, ya sea ascendente o descendente.
- Describir el funcionamiento de algunas técnicas de programación dinámica que se utilizan en el análisis sintáctico porque son capaces de representar ambigüedades.
- Describir el concepto de gramática libre de contexto probabilística y modelar un problema de ambigüedad utilizando este tipo de gramática.
- Aplicar métodos basados en gramáticas libres de contexto probabilísticas para solventar problemas de ambigüedad.

Accede a los ejercicios de autoevaluación a través del aula virtual

5.2. Ambigüedad en el análisis sintáctico

Accede al vídeo «Ambigüedad del análisis sintáctico», «La forma normal de Chomsky», «Reconocimiento sintáctico» a través del aula virtual

Uno de los mayores problemas que se dan en el análisis sintáctico es la ambigüedad. La **ambigüedad estructural** se debe a las múltiples reglas representadas en una gramática, que provienen del uso común de la lengua y que permiten que se puedan encontrar varios resultados del análisis sintáctico que se ajusten correctamente a la analizada.

En una frase en la que aparezca la coordinación «hombres y mujeres mayores», será difícil determinar en el análisis sintáctico si el emisor de la frase se refiere a que tanto los hombres como las mujeres son mayores o, por el contrario, se refiere a todos los hombres, sean o no mayores, y a las mujeres mayores.

- En el primer caso, el adjetivo «mayores» está asociado al sintagma nominal compuesto de la coordinación de los dos nombres: «hombres» y «mujeres». Esta relación se podría escribir utilizando corchetes como [[hombres y mujeres] mayores].

- En el segundo caso, el adjetivo «mayores» está asociado solo al nombre «mujeres» y el sintagma nominal que forman estas dos palabras está relacionado a través de la conjunción copulativa «y» al nombre «hombres». Por tanto, esta relación se podría expresar utilizando la notación de corchetes como [hombres y [mujeres mayores]].

El siguiente ejemplo ilustrativo presenta un caso de ambigüedad estructural en la que existen dos posibles resultados del análisis sintáctico de una oración y que se representan gráficamente en forma de árboles sintácticos. Al final, el analizador sintáctico que obtiene esos dos resultados debe optar por resolver la ambigüedad eligiendo uno de los árboles sintácticos como salida del proceso de análisis.

Ejemplo ilustrativo 1. Ambigüedad en análisis sintáctico de la frase de Groucho Marx en la película *Animal Crackers* (1930): *«I shot an elephant in my pajamas»*.

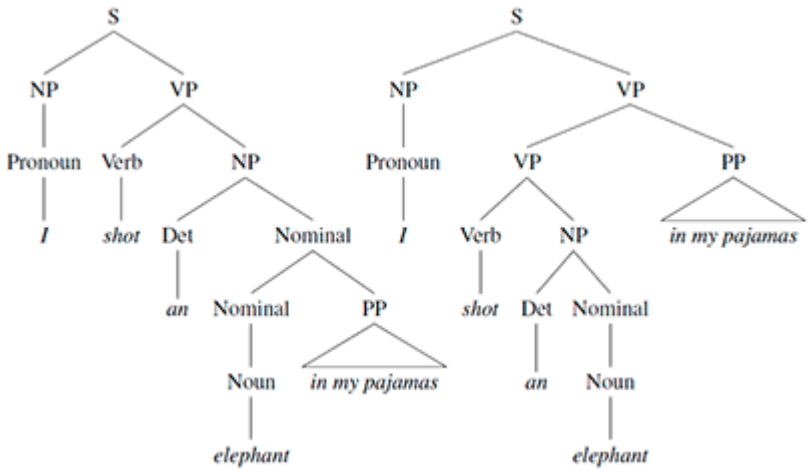


Figura 27. Árboles sintácticos que representan el resultado del análisis sintáctico. Fuente: Jurafsky y Martin, 2009.

El análisis sintáctico de la frase *«I shot an elephant in my pajamas»* puede derivar en dos posibles soluciones y, por tanto, se observa en este caso el problema de la ambigüedad estructural. Tal como se muestra en la figura 1, el elemento *«in my pajamas»* puede ser parte del sintagma nominal (NP) cuyo núcleo es el nombre (*Noun*) *«elephant»*, en el árbol sintáctico de la izquierda, o parte del predicado verbal (VP) cuyo núcleo es el verbo (*Verb*) *«shot»*, en el árbol sintáctico de la derecha.

El problema de la ambigüedad en la estructura de las frases afecta el funcionamiento de la mayoría de los sistemas de procesamiento del lenguaje natural. Entonces, también afecta los analizadores sintácticos que utilizan gramáticas libres de contexto y que se basan en búsqueda, ya sea ascendente o descendente, presentados en el tema anterior. Por lo tanto, esos analizadores deben ser capaces de elegir un único

resultado correcto del análisis de entre la multitud de resultados posibles. La selección del mejor resultado del análisis, es decir, del mejor árbol sintáctico, se realiza normalmente a través de un proceso de desambiguación sintáctica que requiere de fuentes de conocimiento estadístico, semántico y contextual.

Métodos más avanzados evitan el uso de conocimiento lingüístico para realizar la desambiguación *a posteriori* e integran el **proceso de desambiguación** directamente en el análisis sintáctico. Algunos de estos métodos se basan en programación dinámica y otros, en métodos probabilísticos que se presentan en este tema.

Accede a los ejercicios de autoevaluación a través del aula virtual

5.3. Métodos para el análisis sintáctico basados en programación dinámica

Accede al vídeo «Métodos de análisis basados en programación dinámica» a través del aula virtual

Técnicas de **programación dinámica** se utilizan en el análisis sintáctico para tratar el problema de la ambigüedad estructural. La programación dinámica es un método que sirve para optimizar la resolución de problemas que pueden ser discretizados y secuencializados. Así, la programación dinámica permite resolver problemas complejos de una forma más eficiente tratándolos como subproblemas más sencillos.

El principal fundamento de la **programación dinámica** es que las soluciones óptimas de subproblemas permiten encontrar la solución óptima del problema en su conjunto.

Basándose en esta idea, en la programación dinámica se completan de forma sistemática tablas de soluciones para los diferentes subproblemas y, cuando estas tablas están completas, contienen la solución a todos los subproblemas necesarios para resolver el problema global.

La programación dinámica se utiliza en el análisis sintáctico para tratar el problema de la **ambigüedad**. Para ello, los **subproblemas** representan los posibles árboles sintácticos para todos los constituyentes sintácticos detectados como entrada del análisis.

El algoritmo más utilizado para implementar el análisis sintáctico basado en programación dinámica es el **algoritmo de Cocke-Kasami-Younger (CKY)** (Kasami, 1965 y Younger, 1967). El algoritmo CKY se aplica cuando la gramática utilizada en el análisis sintáctico es del tipo *Chomsky Normal Form* (CNF) (Chomsky, 1963). En una gramática CNF las reglas solo pueden tener a la derecha:

- Dos símbolos no terminales (regla $A \rightarrow B\ C$).
- O un símbolo terminal (regla $A \rightarrow w$).

Sin embargo, esto no representa un problema ya que una gramática libre de contexto se puede transformar en una gramática CNF sin perder expresividad.

El primer paso del algoritmo CKY es **convertir la gramática libre de contexto en una gramática CNF**. Para ello, todas aquellas reglas que cumplen por defecto con las condiciones de una CNF (esto es, que a la derecha tengan dos símbolos no terminales o un símbolo terminal) son directamente copiadas a la nueva gramática. El resto de reglas que no cumplen con las condiciones de una gramática CNF son transformadas como se describe a continuación.

Aquellas reglas que mezclan símbolos terminales y no terminales son alteradas con la introducción de un no símbolo terminal comodín que involucra únicamente el símbolo terminal original. Por ejemplo, una regla para un verbo en infinitivo en inglés

como *INF-VP* → *to VP* sería sustituida por las dos reglas *INF-VP* → *TO VP* y *TO* → *to*. Ahora sí, estas nuevas reglas cumplen las condiciones de una gramática CNF.

Las reglas con un solo símbolo no terminal a la derecha, llamémosle **unitarias**, se pueden eliminar reescribiendo la parte derecha de las reglas originales con el lado derecho de todas aquellas reglas no unitarias. Más formalmente, si $A \Rightarrow B$ es una cadena de una o más reglas unitarias y $B \rightarrow \gamma$ es una regla no unitaria, entonces podemos añadir $A \rightarrow \gamma$ para cada regla en la gramática y eliminar todas las reglas unitarias.

Se puede demostrar que esta operación puede conducir a un aplanamiento sustancial de la gramática y la consiguiente promoción de terminales a niveles bastante altos en los árboles resultantes.

Las reglas con más de 2 términos en el lado derecho se normalizan a través de la introducción de nuevos símbolos no terminales que extienden las secuencias más largas en varias reglas nuevas. Formalmente:

$$A \rightarrow B\ C\ \gamma$$

Entonces, si tenemos una regla como la anterior, podemos reemplazar el par más a la izquierda de los símbolos no terminales con un nuevo símbolo no terminal e introducir las siguientes nuevas reglas:

$$A \rightarrow X1\ \gamma$$
$$X1 \rightarrow B\ C$$

Este proceso se puede repetir tantas veces como sea necesario hasta alcanzar reglas de una longitud 2. La elección del par de símbolos no terminales a reemplazar es puramente arbitraria; cualquier procedimiento sistemático que resulte en reglas binarias es adecuado. Una vez se hayan convertido todas las reglas en binarias se añaden a la nueva gramática, finalizando aquí el proceso de conversión a CNF.

Una vez tengamos la gramática en formato CNF, podemos aplicar el algoritmo CKY, el cual nos permite llevar a cabo el proceso de reconocimiento sintáctico. Con nuestra gramática en CNF, cada nodo no terminal por encima del nivel de categoría gramatical en un árbol sintáctico tendrá exactamente dos hijos. Se puede usar una matriz de dos dimensiones para codificar la estructura de todo un árbol.

Para una frase de longitud n, trabajaremos con la parte superior triangular de una matriz:

$$(n + 1)\, x\, (n + 1)$$

Cada celda $[i, j]$ de esta matriz contiene el conjunto de símbolos no terminales que representan todos los constituyentes sintácticos que abarcan posiciones de entrada desde i hasta j. Ya que nuestro sistema de indexación comienza en 0, se deduce que la celda que representa la entrada completa reside en la posición $[0, n]$ de la matriz.

Dado que cada entrada no terminal en nuestra tabla tiene dos hijos en el análisis sintáctico, podemos decir que:

- Para cada constituyente sintáctico representado por una entrada $[i, j]$, tiene que haber una posición en la entrada k, que puede ser dividida en dos partes de tal manera que:

$$i < k < j$$

- Dada una posición k:
  - El primer constituyente sintáctico $[i, k]$ debe estar a la izquierda de la entrada $[i, j]$ en algún lugar a lo largo de la fila i.
  - Y la segunda entrada *[k, j]* debe encontrarse por debajo de aquel, a lo largo de la columna j.

La superdiagonal en la matriz contiene las categorías gramaticales para cada una de las palabras de la oración. Las subsiguientes diagonales por encima de la superdiagonal contienen los constituyentes sintácticos para los diferentes tramos de longitud creciente en la oración.

Vista esta configuración, **el reconocimiento CKY** consiste en rellenar la tabla de análisis sintáctico de la manera correcta. Para ello, se procederá de abajo hacia arriba de manera que, a la hora de rellenar la celda $[i, j]$, las celdas que pueden contener partes que podrían contribuir a esta entrada de la tabla (las celdas a la izquierda y debajo) deben estar ya rellenas.

El pseudocódigo del algoritmo se muestra en la figura 2. Dicho algoritmo rellena la matriz triangular superior operando por columnas de abajo a arriba y de izquierda a derecha como se muestra en la figura 3. Este esquema garantiza que en cada momento tengamos toda la información necesaria.

```
function CKY-PARSE(words, grammar) returns table

for j ← from 1 to LENGTH(words) do
    for all {A | A → words[j] ∈ grammar}
            table[j − 1, j] ← table[j − 1, j] ∪ A
    for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
            for all {A | A → BC ∈ grammar and B ∈ table[i, k] and C ∈ table[k, j]}
                    table[i,j] ← table[i,j] ∪ A
```

Figura 28. Pseudocódigo del algoritmo CKY.
Fuente: Jurafsky y Martin, 2009.

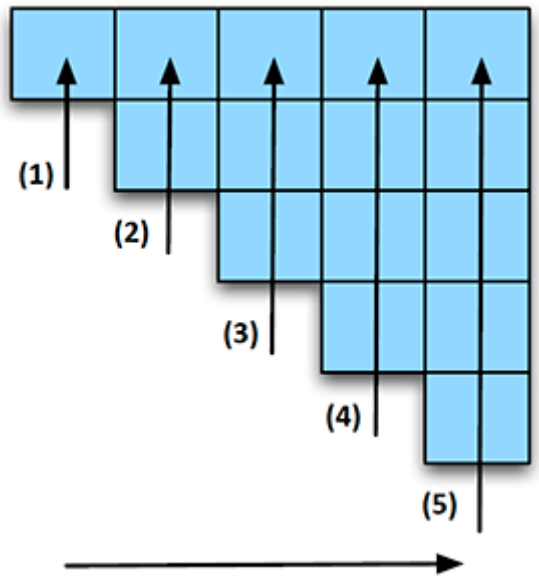


Figura 29. Secuencia seguida para rellenar la tabla en el proceso de reconocimiento CKY.
Fuente: Jurafsky y Martin, 2009.

La figura 4 muestra el caso general de rellenado de la celda $[i, j]$. En cada partición, el algoritmo considera si los contenidos de las dos celdas pueden ser combinados de modo que se cumpla alguna de las reglas de la gramática.

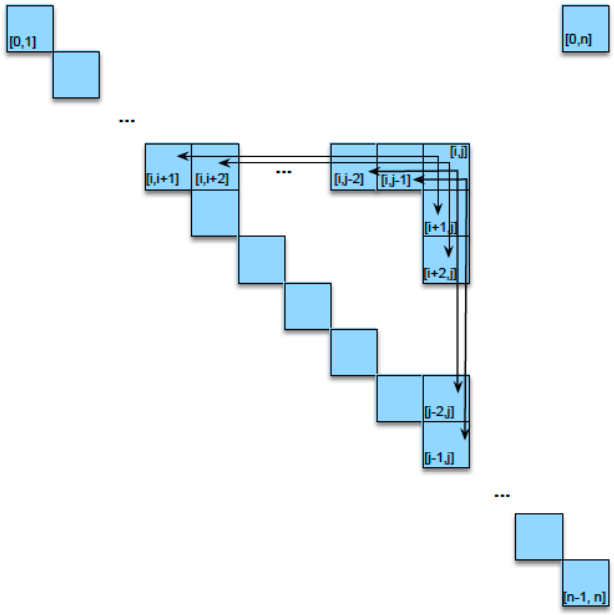


Figura 30. Todas las posibles formas de rellenar la celda $[i, j]$ en la tabla CKY.
Fuente: Jurafsky y Martin, 2009.

Ejemplo ilustrativo 2. Algoritmo CKY para el reconocimiento sintáctico de la frase en inglés *«Book the flight through Houston»* (en español, «Reserva el vuelo a través de Houston»).

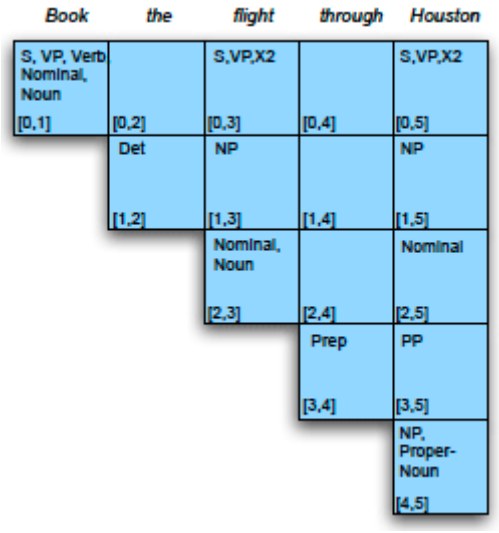


Figura 31. Matriz del análisis sintáctico al aplicar el algoritmo CKY.
Fuente: Jurafsky y Martin, 2009.

La figura 5 muestra la matriz del análisis sintáctico completo para la frase *«Book the flight through Houston»* cuando se utiliza la gramática en formato CNF presentada en la figura 6 para realizar el reconocimiento sintáctico.

$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Lexicon

Det → *that* | *this* | *the* | *a*
Noun → *book* | *flight* | *meal* | *money*
Verb → *book* | *include* | *prefer*
Pronoun → *I* | *she* | *me*
Proper-Noun → *Houston* | *NWA*
Aux → *does*
Preposition → *from* | *to* | *on* | *near* | *through*

Figura 32. Gramática libre de contexto (columna izquierda) y en formato CNF (columna derecha) y lexicón (abajo) para realizar el análisis sintáctico en lengua inglesa. Fuente: Jurafsky y Martin, 2009.

El ejemplo concreto de cómo se rellenarían las celdas de la columna 5 de la matriz después de leer la palabra *«Houston»* se muestra en la figura 7. Las flechas señalan los elementos de las dos celdas que se utilizan para agregar una entrada a la tabla.

Cabe destacar que en la celda [0; 5] se indica la presencia de tres posibles análisis alternativos para esta entrada:

- Uno, donde la preposición (PP) modifica la palabra *«flight»*.
- Otro, donde esta misma preposición modifica la palabra *«book»*.
- Y el tercero, que captura la regla *VP* → *X2 PP*.

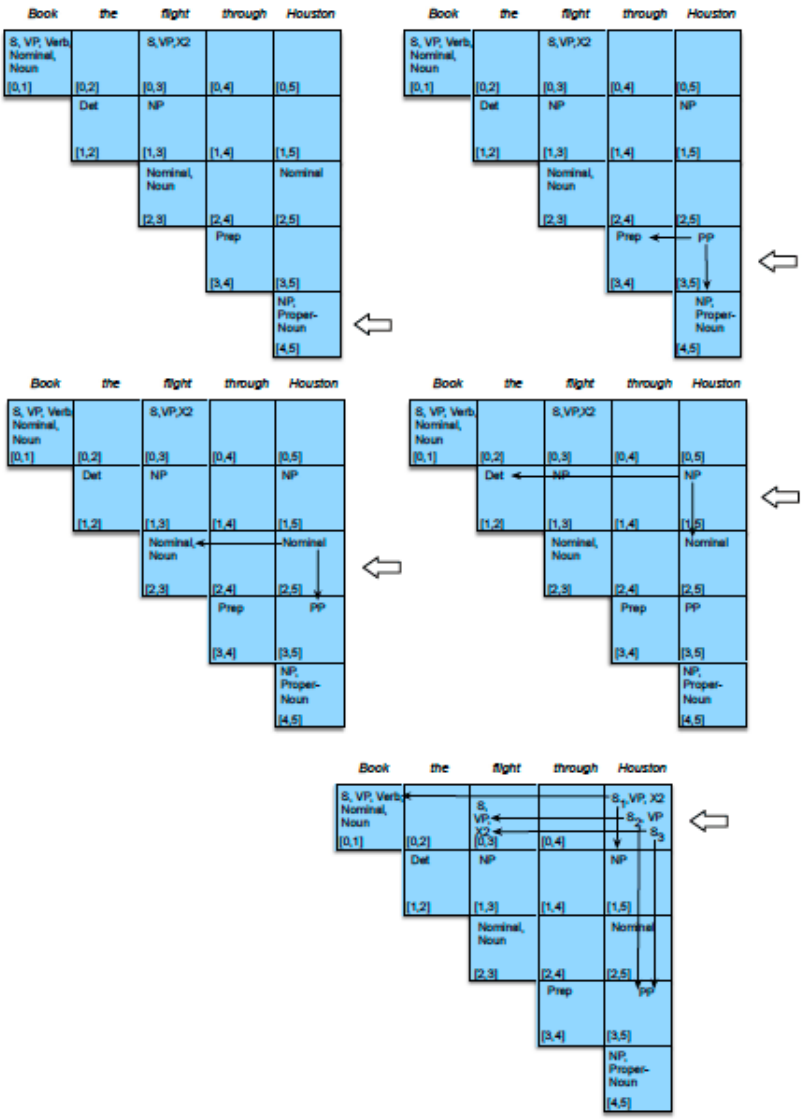


Figura 33. Rellenado de las celdas de la columna 5
después de leer la palabra *«Houston»*.
Fuente: Jurafsky y Martin, 2009.

Finalmente, llegamos a la **fase de análisis sintáctico**. Hay que tener en cuenta que el algoritmo mostrado en la figura 3 es un reconocedor sintáctico, no un analizador sintáctico. Por tanto, para convertirlo en un analizador capaz de devolver todos los análisis sintácticos posibles para una entrada dada, podemos hacer dos simples cambios en el algoritmo.

3. El primer cambio es aumentar las entradas de la tabla de manera que cada símbolo no terminal esté emparejado con punteros a las

entradas de la tabla de las que se derivaron (parecido a como se muestra en la figura 7 del ejemplo ilustrativo 2).

4. El segundo cambio consiste en permitir múltiples versiones del mismo símbolo no terminal para ser introducidos en la tabla (véase la figura 7).

Con estos cambios, el cuadro completo contiene todos los posibles análisis sintácticos para una entrada dada. Devolver un único análisis sintáctico arbitrario consiste en la elección de una oración S de la celda $[0, n]$ y luego recuperar de forma recursiva sus constituyentes sintácticos de la tabla.

Accede a los ejercicios de autoevaluación a través del aula virtual

5.4. Métodos probabilistas en el análisis sintáctico

Accede al vídeo «Métodos de análisis probabilístico» a través del aula virtual

La ambigüedad estructural se puede modelar de forma eficiente utilizando el algoritmo CKY presentado en el apartado anterior. Sin embargo, el analizador sintáctico va a devolver múltiples resultados del análisis y para poder resolver la ambigüedad y llegar a una única solución es necesario implementar métodos probabilistas en él. De hecho, la mayoría de los analizadores sintácticos utilizados hoy en día implementan métodos probabilistas como método de **desambiguación**.

Los **analizadores sintácticos probabilistas** calculan la probabilidad de cada posible interpretación del análisis sintáctico y escogen la más probable.

Formalmente, un analizador sintáctico probabilista tiene como objetivo producir el análisis sintáctico más probable $\hat{T}$ para una oración dada S:

$$\hat{T}(S) = \underset{T s.t. S=yield(T)}{\operatorname{argmax}} P(T)$$

Por lo tanto, el analizador sintáctico probabilista va a computar el árbol sintáctico T que maximice $P(T)$, esto es, la probabilidad de dicho árbol. Dicho de otra forma, el analizado sintáctico va a buscar el árbol sintáctico más probable. Notar que la cadena de palabras S se denomina el *«yield»* de cualquier árbol sintáctico que se puede aplicar a la oración a analizar.

Para calcular la **probabilidad de un árbol sintáctico** $P(T)$ de una oración S a partir de las probabilidades de las reglas de una gramática, se considera que las reglas son independientes y se calcula la probabilidad del árbol multiplicando las probabilidades de cada una de las reglas involucradas en el análisis de la oración.

Para calcular la **probabilidad asociada a una regla** representada en una gramática libre de contexto existen dos maneras. La forma más sencilla es utilizar un treebank (Marcus, Santorini y Marcinkiewicz, 1993), esto es, un corpus de frases ya analizados. Dado un treebank, podemos calcular la probabilidad de cada expansión de un símbolo no terminal mediante el recuento del número de veces que se produce la expansión se produce y luego normalizando, tal como se muestra en la siguiente fórmula:

$$P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

Si no tenemos un treebank, pero sí tenemos un analizador sintáctico (no probabilístico), podemos generar los recuentos que necesitamos para el cálculo de las probabilidades asociadas a cada regla. Para ello, analizamos sintácticamente el corpus de frases con el analizador. Si las frases no son ambiguas, sería tan simple como analizar todo el corpus, incrementar un contador para cada regla dada en el análisis sintáctico y, luego, normalizar para obtener las probabilidades.

Como la mayor parte de las oraciones son ambiguas, es decir, tienen múltiples análisis sintácticos, tenemos que mantener una cuenta separada para cada análisis sintáctico de una oración y ponderar cada uno de estos recuentos parciales por la probabilidad del análisis sintáctico en el que aparece. Pero para obtener las probabilidades para ponderar las reglas, tenemos que tener ya un analizador sintáctico probabilístico, lo cual nos lleva a un problema recurrente.

La forma de resolver este problema es mejorar de forma creciente nuestras estimaciones:

- Comenzar con un analizador sintáctico con probabilidades iguales para cada regla.
- Seguidamente, analizar la frase.
- Calcular la probabilidad para cada análisis sintáctico.
- Utilizar estas probabilidades para ponderar los contadores.
- Reestimar las probabilidades de cada regla, y así sucesivamente, hasta que nuestras probabilidades converjan.

El algoritmo estándar para el cálculo de esta solución se llama *inside-outside* (Baker, 1979) y es un caso especial del algoritmo *Expectation-Maximization* (Manning y Schütze, 1999).

Algoritmo CKY probabilístico

La mayoría de los analizadores sintácticos probabilistas modernos se basan en el **algoritmo CKY probabilístico** (Ney, 1991), una versión probabilística del algoritmo CKY presentado en la sección anterior.

El algoritmo CKY probabilístico extiende el algoritmo CKY para incluir información probabilística. La versión probabilística del algoritmo CKY, igual que el algoritmo CKY básico, utiliza una gramática CNF (*Chomsky Normal Form*). Sin embargo, en la versión probabilística, cada regla está anotada con la probabilidad de que se cumpla dicha regla.

1. Por lo tanto, el primer paso del algoritmo CKY probabilístico consiste en convertir las reglas de una gramática libre de contexto anotada con probabilidades al **formato CNF**. Para realizar este proceso se aplica un método análogo al utilizado para el algoritmo CKY básico. En la conversión de las reglas al formato CNF, además de aplicar los criterios explicados en la sección anterior, se deben recalcular las probabilidades de modo que la probabilidad asociada a cada posible árbol resultante del análisis sintáctico de la oración permanezca constante para la nueva gramática CNF.

2. En el segundo paso del algoritmo CKY probabilístico, y una vez se tienen las reglas en formato CNF, se crearía una **matriz** similar a la del CKY básico, pero con una dimensión adicional. Recordemos que en el CKY básico cada celda de la matriz contenía una lista con los constituyentes sintácticos para cada palabra. En el caso CKY probabilístico, cada celda tiene una dimensión adicional que se utiliza para indexar cada constituyente sintáctico. Específicamente, para una frase de longitud n (palabras) y una gramática que contiene V símbolos no terminales (constituyentes sintácticos), tendríamos una matriz $(n + 1)\, x\, (n + 1)\, x\, V$. El valor de cada una de las celdas corresponde en este caso a la probabilidad de cada constituyente o símbolo no terminal para cada palabra.

El pseudocódigo del algoritmo CKY probabilístico se presenta en la figura 8.

```
function PROBABILISTIC-CKY(words, grammar) returns most probable parse
                                                and its probability
  for j ← from 1 to LENGTH(words) do
    for all { A | A → words[j] ∈ grammar}
      table[j − 1, j, A] ← P(A → words[j])
    for i ← from j − 2 downto 0 do
      for k ← i + 1 to j − 1 do
          for all { A | A → BC ∈ grammar,
                        and table[i, k, B] > 0 and table[k, j, C] > 0 }
            if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
              table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
              back[i,j,A] ← {k, B, C}
  return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figura 34. Pseudocódigo del algoritmo CKY probabilístico.
Fuente: Jurafsky y Martin, 2009.

Ejemplo ilustrativo 3. Algoritmo CKY probabilístico para el reconocimiento sintáctico de la frase en inglés *«The flight includes a meal»*, en español significa «El vuelo incluye comida».

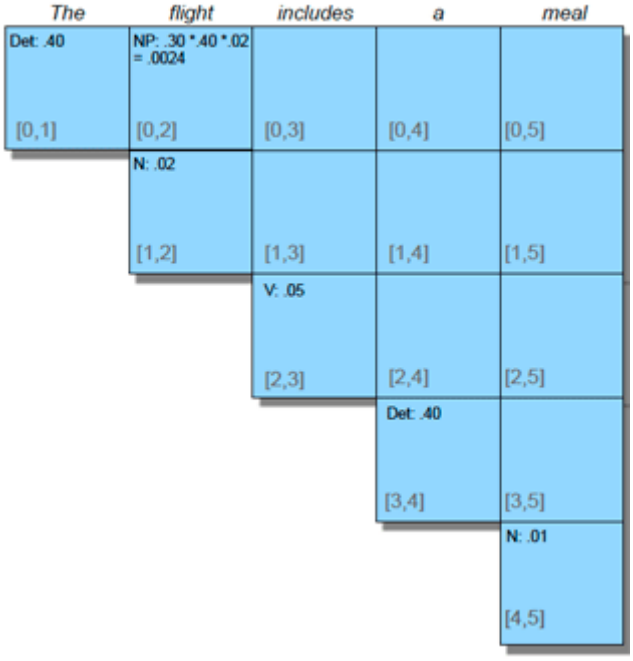
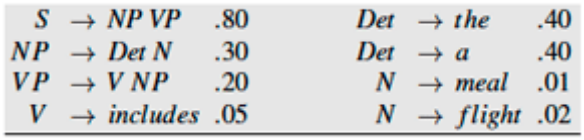


Figura 35. Matriz para el análisis sintáctico aplicando el algoritmo CKY probabilístico.
Fuente: Jurafsky y Martin, 2009.

Los primeros pasos de aplicar el algoritmo probabilístico CKY para analizar la frase *«The flight includes a meal»*, se muestran en la matriz de la figura 9. La gramática en formato CNF que incluye las probabilidades de cada una de las reglas y que se

utiliza para realizar el reconocimiento sintáctico se presenta en la figura 10.



S	$\rightarrow$	$NP\ VP$	.80	Det	$\rightarrow$	*the*	.40
NP	$\rightarrow$	$Det\ N$	.30	Det	$\rightarrow$	*a*	.40
VP	$\rightarrow$	$V\ NP$	.20	N	$\rightarrow$	*meal*	.01
V	$\rightarrow$	*includes*	.05	N	$\rightarrow$	*flight*	.02

Figura 36. Gramática de la lengua inglesa en formato CNF donde cada regla está anotada con la probabilidad y que se utiliza para realizar el análisis sintáctico. Fuente: Jurafsky y Martin, 2009.

Accede a los ejercicios de autoevaluación a través del aula virtual

5.5. Referencias bibliográficas

Baker, J. K. (1979). Trainable grammars for speech recognition. En D. H. Klatt y J. J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America* (pp. 547-550). Nueva York, Estados Unidos: Acoustical Society of America.

Chomsky, N. (1963). Formal properties of grammars. En R. D. Luce, R. Bush y E. Galanter (Eds.), *Handbook of Mathematical Psychology* (Vol. 2, pp. 323-418). Cambridge, Estados Unidos: Wiley.

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics*. New Jersey, Estados Unidos: Prentice-Hall.

Kasami, T. (1965). *An efficient recognition and syntax analysis algorithm for context-free languages*. (Informe No. R-257), Urbana-Champaing, Estados Unidos: Universidad de Illinois.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, Estados Unidos: The MIT Press.

Marcus, M. P., Santorini, B. y Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics, 19*(2), 313-330.

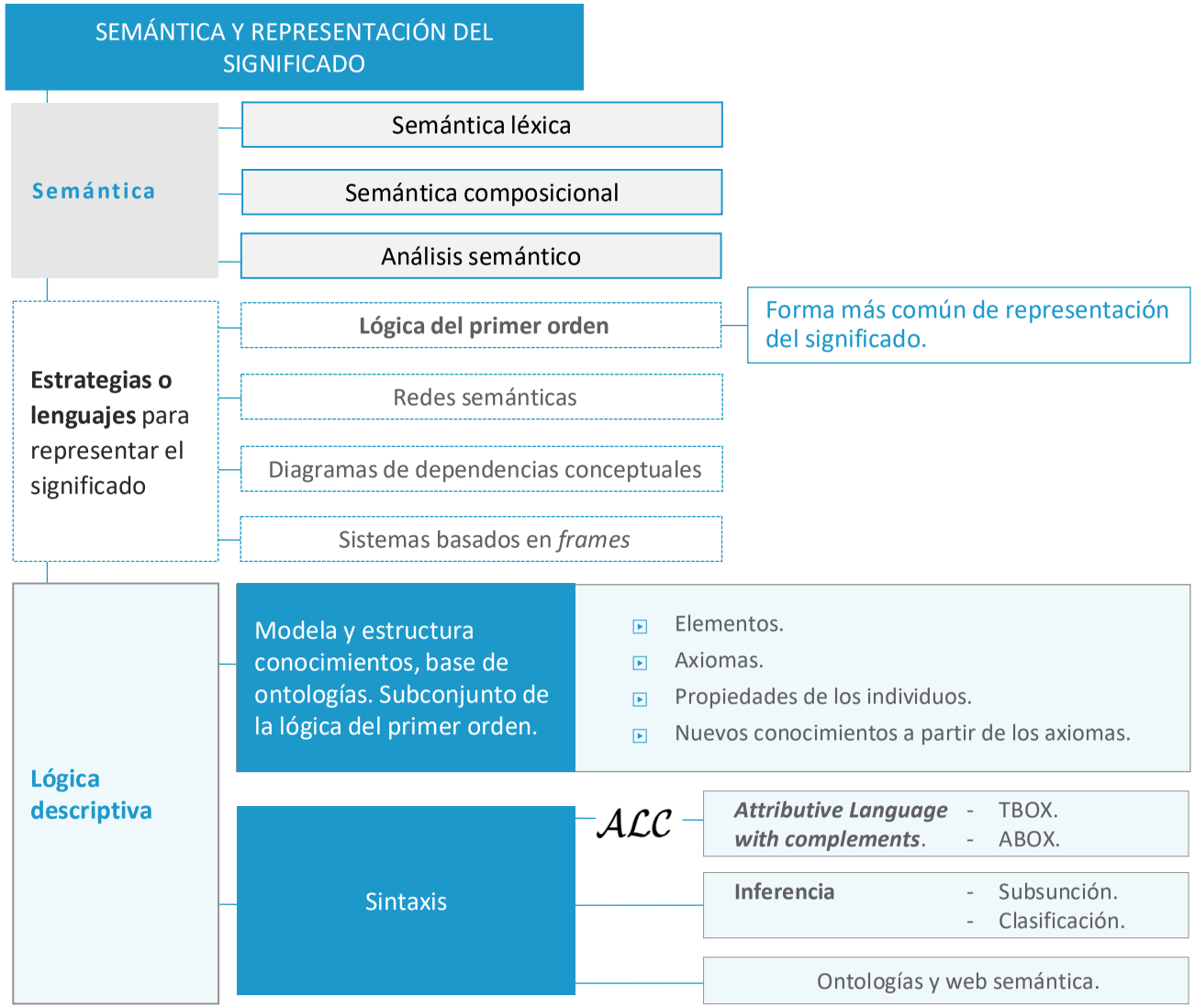
Ney, H. (1991). Dynamic programming parsing for contextfree grammars in continuous speech recognition. *IEEE Transactions on Signal Processing, 39*(2), 336-340.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10, 189-208.

Accede al vídeo «Resumen» a través del aula virtual

Tema 6. Semántica y representación del significado

Índice



Esquema

Ideas clave

6.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Los apartados de los que consta este tema son:

- Semántica.
- Estrategias o lenguajes para representar el significado.
- Lógica de primer orden.
- Lógica descriptiva.
- Referencias bibliográficas.

Al finalizar el estudio de este tema serás capaz de:

- Entender la necesitad de representar el significado del lenguaje natural y describir los requisitos que debe tener una representación del significado.
- Identificar las tres estrategias más básicas para representar el significado: la lógica de primer orden, las redes semánticas y los *frames*.
- Describir la lógica de primer orden, la forma más común de representar el significado.
- Describir la lógica descriptiva, una estructura de representación que sirve para modelar dominios de conocimiento y que es la base de las ontologías.

Accede a los ejercicios de autoevaluación a través del aula virtual

6.2. Semántica

Accede al vídeo «Semántica» a través del aula virtual

La Real Academia Española (RAE, s. f.) en su diccionario de la lengua española define la palabra **semántica:** «Del gr. *σημαντικός sēmantikós* "significativo". 3. f. Ling. Disciplina que estudia el significado de las unidades lingüísticas y de sus combinaciones».

En algunas tareas que requieren procesamiento del lenguaje natural es necesario conocer el **significado de las palabras**, eso quiere decir tener no solo el conocimiento lingüístico (por ejemplo, la información morfológica o sintáctica sobre la palabra), sino que también la información no lingüística sobre esa palabra, esto es, algún tipo de conocimiento del mundo real relacionado con esa palabra.

Si se coge como ejemplo un agente conversacional que permite reservar restaurantes y que realiza recomendaciones basadas en el menú de cada uno de los restaurantes, no será suficiente realizar un análisis morfológico o sintáctico de la petición realizada por el usuario («reserva un restaurante vegetariano»), sino que, además, será necesario tener conocimiento no lingüístico sobre los diferentes tipos de comida para saber a qué concepto del mundo real se hace referencia cuando se habla de comida vegetariana y, así, no recomendar un restaurante donde el menú esté compuesto solo de platos de carne.

La **semántica léxica** «estudia el significado de las palabras, así como las diversas relaciones de sentido que se establecen entre ellas» según su definición en el diccionario de la lengua española de la RAE (s. f.).

Un análisis en profundidad de la semántica léxica, incluyendo la descripción detallada del significado de las palabras y de las relaciones entre significados, se presenta en capítulos posteriores.

Conocer el **significado de una oración**, y no solo el significado de una palabra, incluyendo cualquier tipo de conocimiento no lingüístico de la oración es también imprescindible para realizar eficientemente diversas tareas de procesamiento del lenguaje natural.

La **semántica composicional** «estudia el significado de los sintagmas y las oraciones» según su definición en el diccionario de la lengua española de la RAE (s. f.).

El **análisis semántico**, que se va a estudiar posteriormente, tiene como objetivo producir una representación del significado de una oración. La creación automática de una representación rigurosa del significado de la oración requiere de múltiples fuentes de conocimiento y de técnicas de inferencia. Entre las fuentes de conocimiento necesarias destaca el conocimiento sobre:

- Los significados de las palabras.
- Los significados asociados a las construcciones gramaticales.
- La estructura del discurso.
- El tema en cuestión.
- El estado de las cosas en el momento en el que tiene lugar el discurso.

Por lo tanto, en el procesamiento del lenguaje natural existen múltiples razones que confirman la necesitad de representar el significado de una manera formal.

Accede a los ejercicios de autoevaluación a través del aula virtual

6.3. Estrategias o lenguajes para representar el significado

Accede al vídeo «Lenguajes de representación del significado» a través del aula virtual

Una representación formal del significado debe ser verificable, inequívoca, expresiva y permitir la inferencia de nuevo conocimiento. La mayoría de los modelos utilizados en las representaciones del significado tienen en común la capacidad de representar objetos, sus propiedades y las relaciones entre estos.

Existen múltiples estrategias o lenguajes que sirven para representar el significado, pero comentaremos las siguientes:

Figura 37. Estrategias o lenguajes para representar el significado.

La **lógica de primer orden** es un lenguaje formal que permite la representación del conocimiento y satisface todas las necesidades que tiene que cumplir un modelo para poder representar correctamente el significado. El lenguaje de la lógica de primer orden está constituido por objetos y relaciones que se expresan formalmente tal como se explica en la siguiente sección.

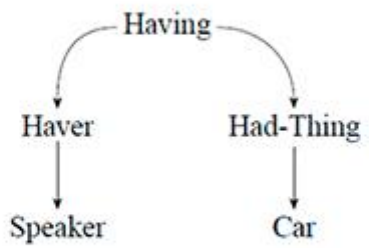
Las **redes semánticas** proporcionan una representación gráfica del significado. En estas redes, los objetos se representan como los nodos del gráfico y las relaciones entre los objetos se representan mediante enlaces entre los nodos a los que se les asigna un nombre.

Los **diagramas de dependencias conceptuales** proporcionan una representación gráfica de las dependencias conceptuales (Schank, 1972). El modelo de estas permite representar el significado basado en descomponer las oraciones en primitivas que representan las acciones, los participantes en las acciones o actores y los objetos.

Los **sistemas basados en *frames*** representan el significado a través de plantillas que definen la estructura del modelo (Fillmore, 1985). En los sistemas basados en *frames*, los objetos se representan como estructuras de características, a las cuales se las llaman *slots* y los valores que pueden tomar estos huecos *(slots*) pueden ser valores atómicos u otros *frames* embebidos. Los sistemas basados en *frames* también pueden ser representados como gráficos.

Ejemplo ilustrativo 1. Diferentes representaciones del significado de la oración *«I have a car»*.

$$\exists e,y\ Having(e) \wedge Haver(e,Speaker) \wedge HadThing(e,y) \wedge Car(y)$$



Car
⇑ POSS-BY
Speaker

```
Having
 Haver:   Speaker
 HadThing:  Car
```

Figura 38. Cuatro representaciones diferentes del significado de *«I have a car»*.
Fuente: Jurafsky y Martin, 2009.

La primera fila muestra la oración representada con lógica de primer orden; el gráfico en el centro es un ejemplo de red semántica; la última fila a la izquierda presenta un diagrama de dependencias conceptuales y, a la derecha, se muestra una representación basada en *frames*.

Una última estrategia para representar el significado y que permite aunar el poder formal de la lógica de primer orden con la visión más intuitiva de las redes semánticas es la **lógica descriptiva**. La gran relevancia de esta en la representación del significado deriva de que es la base formal sobre la que se desarrolla la web semántica.

Accede a los ejercicios de autoevaluación a través del aula virtual

6.4. Lógica de primer orden

Accede al vídeo «Lógica descriptiva» a través del aula virtual

Para completar el estudio de esta sección deberás leer el apartado **8.2. Sintaxis y semántica de la lógica de primer orden,** en las **páginas 277 a 286** del libro: Russell, S. J. y Norvig, P. (2004). *Inteligencia Artificial: un enfoque moderno*. Madrid: Pearson Educación.
Disponible en la Biblioteca Virtual de UNIR.

La lógica de primer orden es la forma más común de representar el significado por tener una sintaxis y una semántica del lenguaje construida sobre objetos y relaciones. Además, este lenguaje formal permite la inferencia de conocimiento.

Accede a los ejercicios de autoevaluación a través del aula virtual

6.5. Lógica descriptiva

Accede al vídeo «Lógica descriptiva» a través del aula virtual

La **lógica descriptiva** no se refiere a una lógica concreta ni a un formalismo concreto, sino a una familia completa de lógicas que cumplen con unas determinadas características. Es decir, hay diferentes lógicas descriptivas, cada una con sus características, pero todas ellas con algunos puntos en común.

En general, las lógicas descriptivas contienen los siguientes **elementos**:

- Un mecanismo para generar descripciones de conceptos. Nótese que esto es de especial importancia, como nos recuerda el propio nombre de este tipo de lógicas.
- Un mecanismo para introducir axiomas, sobre los que se apoyan las descripciones.
- Un mecanismo para introducir propiedades de los individuos.
- Un método para inferir nuevos conocimientos a partir de los axiomas.

En lo referente a la **sintaxis**, la lógica descriptiva se basa en notaciones diseñadas para hacer más fácil **describir definiciones y propiedades** de las categorías (o clases) de objetos, a diferencia de la sintaxis de la lógica de primer orden, que pretende facilitar las afirmaciones sobre objetos. Es por eso por lo que las lógicas descriptivas eliminan la necesidad de utilizar una variable para expresar el sujeto, ya que este va implícito en el axioma.

Ejemplo ilustrativo 2. Diferencias en la sintaxis de la lógica descriptiva en relación con la lógica de primer orden.

En la lógica de primer orden se expresa que un hombre es feliz si tiene salud, dinero y amor de la siguiente forma:

$$HombreFeliz(x) \Leftrightarrow TieneSalud(x) \wedge TieneDinero(x) \wedge TieneAmor(x)$$

Es decir que la variable `x` se utiliza para indicar cualquier individuo que cumple las condiciones de ser feliz y, por tanto, de tener salud, tener dinero y tener amor.

En el lenguaje CLASSIC (Borgida, Brachman, McGuinness, y Alperin Resnick, 1989), que es un ejemplo típico de lógica descriptiva, la misma sentencia se escribiría de la siguiente forma:

```
HombreFeliz = Y(TieneSalud, TieneDinero, TieneAmor))
```

Así, se observa la novedad que aportan las lógicas descriptivas en relación a las lógicas de primer orden al eliminar la necesidad de una variable para expresar el sujeto.

Además, las lógicas descriptivas permiten efectuar operaciones lógicas directamente en los predicados, en vez de tener que crear primero oraciones que se unen mediante conectores como sería el caso de la lógica de primer orden. Así, la sintaxis de las lógicas descriptivas tiene una legibilidad mayor que la de las lógicas de primer orden. Esto permite la construcción de expresiones combinadas más complejas que, igualmente, se podrían expresar en el equivalente de lógica de primer orden, pero darían como resultado una expresión más difícil de manejar.

La lógica descriptiva puede considerarse un **subconjunto de la lógica de primer orden** (Baader, Horrocks y Sattler, 2009). El hecho de que sea un subconjunto hace que sea más manejable y también que algunas operaciones sean más sencillas y eficientes. Sin embargo, esto tiene sus problemas: las lógicas descriptivas no permiten, como norma general, realizar descripciones basadas en la negación ni en la disyunción.

Existen diferentes variantes de sintaxis para las lógicas descriptivas, siendo $\mathcal{ALC}$ (*Attributive Language with Complements*) la sintaxis más extendida.

Una lógica descriptiva estructura el conocimiento en dos cajas (*boxes*):

- El TBOX es la parte del conocimiento que se refiere a la definición de las clases, las relaciones y las propiedades.

- El ABOX es la parte del conocimiento en la que se nombran los individuos pertenecientes a las clases y se definen las relaciones y propiedades que tienen estos individuos.

Ejemplo ilustrativo 3. TBOX y ABOX para la descripción de conocimiento sobre las relaciones de parentesco utilizando una sintaxis $\mathcal{ALC}$.

El TBOX contiene las definiciones de los diferentes tipos de familiares y las diferentes relaciones de parentesco:

Mujer ≡ Persona _ SexoFemenino

Hombre ≡ Persona _ ¬Mujer

Madre ≡ Mujer _ ∃tieneHijo.Persona

Padre ≡ Hombre _ ∃tieneHijo.Persona

PadreOMadre ≡ Padre _ Madre

Abuela ≡ Madre _ ∃tieneHijo.PadreOMadre

MujerConMuchosHijos ≡ Mujer _ >3 tieneHijo

MadreSinHija ≡ Madre _ ∀tieneHijo.¬ Mujer)

En el ABOX se definen los individuos que son miembros de las clases y se establecen las relaciones entre los individuos:

MadreSinHija(Elena)

Padre(Pedro)

tieneHijo(Elena, Pedro)

tieneHijo(Pedro,Lucas)

tieneHijo(Elena, Javier)

La lógica descriptiva tiene como cometido principal realizar tareas de **inferencia**, tales como la subsunción y la clasificación.

La subsunción es la comprobación de si una categoría es un subconjunto de otra, a través de la comparación de sus definiciones.

La clasificación es la comprobación de si un objeto pertenece a una categoría. Además, algunas lógicas descriptivas también permiten verificar la **consistencia** de la definición de una categoría, es decir comprobar si el criterio de pertenencia puede ser satisfecho lógicamente.

La lógica descriptiva tiene especial importancia porque proporciona el formalismo lógico necesario para diseñar **ontologías** y la **web semántica**. De hecho, OWL (*Web Ontology Language*), el lenguaje para publicar y compartir datos usando ontologías a través de la web, está basado en la lógica descriptiva.

Accede a los ejercicios de autoevaluación a través del aula virtual

6.6. Referencias bibliográficas

Baader, F., Horrocks, I. y Sattler, U. (2009). Description logics. En *Handbook on ontologies*. Heidelberg, Alemania: Springer.

Borgida, A., Brachman, R. J., McGuinness, D. L. y Alperin Resnick, L. (1989). Classic: a structural data model for objects. *ACM SIGMOD Record, 18*(2), 58-67.

Fillmore, C. J. (1985). Frames and the semantics of understanding. *Quaderni di Semantica, 6*(2), 222-254.

RAE. (S. f.). Semántica. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=XVRDns5

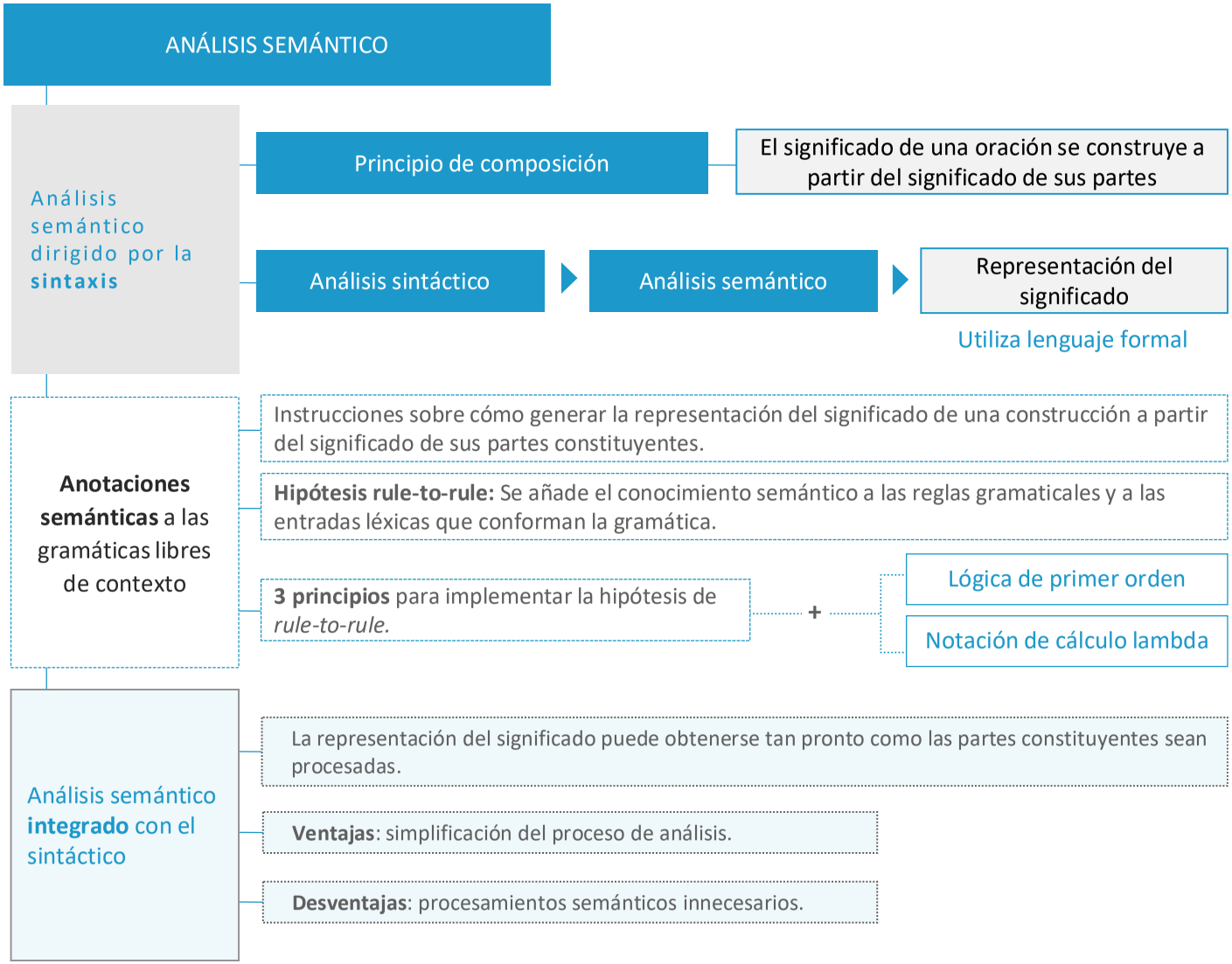
Schank, R. C. (1972). Conceptual dependency: A theory of natural language processing. *Cognitive Psychology*, 3, 552-631.

Accede al vídeo «Resumen» a través del aula virtual

Tema 7. Análisis semántico

Índice

Esquema



Ideas clave

7.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Los apartados de los que consta este tema son:

- Análisis semántico dirigido por la sintaxis.
- Anotaciones semánticas a las gramáticas libres de contexto.
- Análisis semántico integrado con el análisis sintáctico.
- Referencias bibliográficas.

Al finalizar el estudio de este tema serás capaz de:

- Describir el funcionamiento del análisis semántico basado en el principio de composición del significado, es decir, el análisis semántico dirigido por la sintaxis.
- Modelar las anotaciones semánticas que se hace de las gramáticas libres de contexto para que la información sintáctica representada en forma de reglas se complemente con información semántica.
- Describir las características de realizar el análisis semántico en paralelo con el análisis sintáctico, al tener la semántica incorporada en la representación sintáctica.

7.2. Análisis semántico dirigido por la sintaxis

Accede al vídeo «Análisis semántico dirigido por la sintaxis» a través del aula virtual

El objetivo del análisis semántico es producir una representación del significado de la oración y, tal como se ha explicado en el tema anterior, requiere de múltiples fuentes de conocimiento como, por ejemplo, el conocimiento sobre los significados de las palabras. Entonces, el análisis semántico dirigido por la sintaxis se basa en el **principio de composición.**

La idea fundamental del principio de composición es que el significado de una oración se construye a partir del significado de sus partes.

Una frase se compone de palabras y las palabras son las unidades básicas de significado. Por lo tanto, se podría interpretar que, de acuerdo con el principio de composición, el significado de una oración es el conjunto de los significados de las palabras que la componen, sin embargo, esta primera interpretación tan básica es poco útil.

El significado de una oración no se basa solamente en el significado de las palabras que la componen, sino que también en el orden y la agrupación de palabras y en las relaciones entre las palabras en la frase. Es decir, que el significado de una frase está también basado en su **estructura sintáctica**. Por lo tanto, en el análisis semántico dirigido por la sintaxis, la composición del significado de la oración se hace a partir de la estructura sintáctica de la frase.

La figura 1 muestra una posible **estructura del proceso de análisis semántico dirigido por la sintaxis**. Se observa que el resultado del análisis sintáctico de una oración sirve como entrada para el análisis semántico. De esta forma, una frase que vaya a ser analizada pasa primero por un analizador sintáctico que extraiga la información sintáctica relativa a las relaciones estructurales entre palabras. La estructura sintáctica extraída, que normalmente se representa como un árbol sintáctico tal como se ha visto en temas anteriores, sirve para alimentar el analizador semántico, que producirá una representación del significado de la oración.

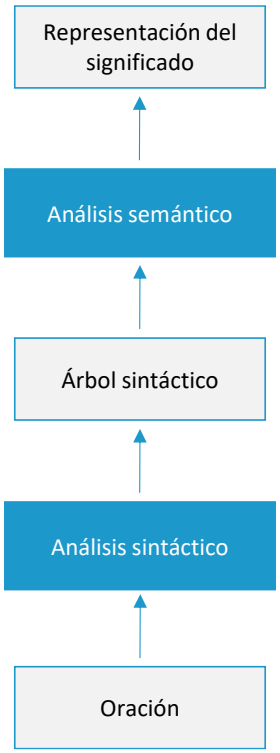


Figura 39. Estructura del proceso de análisis semántico dirigido por la sintaxis.

Cabe destacar que, en diferentes tareas de PLN presentadas en temas anteriores, se ha observado el problema de la ambigüedad. Sin embargo, en el análisis semántico dirigido por la sintaxis se asume que la **ambigüedad** no es un problema para el analizador semántico, ya que se va a tratar en otras etapas del proceso de análisis.

Por ejemplo, el analizador sintáctico será capaz de resolver las ambigüedades en los árboles sintácticos y solo proporcionará el mejor árbol sintáctico al analizador semántico. Otra opción sería que el analizador sintáctico encontrara varios posibles árboles sintácticos y se encargara de alimentar al analizador semántico con cada uno de estos posibles árboles y, una vez obtenidas las diferentes representaciones del significado, se aplicara un proceso de desambiguación en un nivel superior.

> **Ejemplo ilustrativo 1.** Análisis semántico dirigido por la sintaxis de la frase en inglés *«Francis likes Frasca»*.

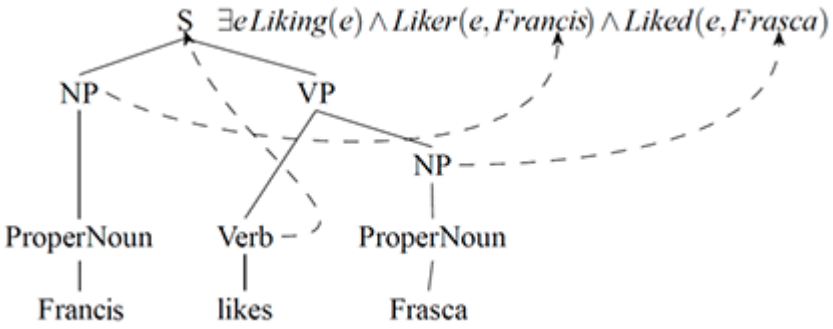


Figura 40. Árbol sintáctico anotado con la representación del significado resultante del análisis semántico de *«Francis likes Frasca»*.
Fuente: Jurafsky y Martin, 2009.

El análisis semántico de la frase *«Francis likes Frasca»* (en español, «A Francis le gusta Frasca») se realiza a partir del árbol sintáctico derivado del proceso previo de análisis sintáctico. Al árbol se le asigna la representación del significado utilizando un lenguaje formal como, por ejemplo, lógica de primer orden.

Las líneas discontinuas en la figura 2 muestran la asociación de los nodos del árbol sintáctico con los diferentes elementos de la representación del significado a los que hacen referencia. Por ejemplo:

«Frasca» es nombre propio (*ProperNoun*), núcleo del sintagma nominal (NP) y forma parte del predicado verbal (VP), cuyo núcleo es «*likes*», el verbo (*Verb*); en el significado de la frase se representa, en lógica de primer orden, como el objeto de la expresión «*Liked(e,Frasca)*». Es decir, «Frasca» es el objeto de una relación del tipo *Liked,* donde el sujeto es el elemento *e,* del cual se sabe que es del tipo *Liking* y que, además, está relacionado con el objeto «Francis» a través de la relación *Liker*.

Para poder obtener e interpretar la representación del significado, es necesario definir formalmente utilizando lógica de primer orden todo el conocimiento necesario, es decir, qué significa una relación del tipo *Liked*, un elemento del tipo *Liking* o una relación del tipo *Liker*.

Como resultado del análisis semántico se obtiene la **representación del significado de la oración**, y esta representación se modela utilizando un lenguaje formal como puede ser la lógica de primer orden o la lógica descriptiva. Por tanto, el analizador

semántico va a necesitar acceder a una descripción formal del conocimiento sobre el significado de las palabras con las que trabaja.

Esto se observa en el ejemplo ilustrativo 1 donde se ha modelado el verbo *«like»* (gustar) a través del objeto *Liking* y las relaciones *Liked* y *Liker*. El analizador semántico, al encontrarse en el árbol sintáctico con el verbo «like», debe saber proporcionar la representación adecuada para ese verbo en concreto.

El algoritmo del analizador semántico interpreta y conoce cierta información general sobre los árboles sintácticos como que, entre otras cosas:

- Es el verbo el que carga con el significado principal de la oración.
- Los sintagmas que complementan el verbo son elementos que proporcionan información relacionada con el verbo.

Esta información es genérica, sin embargo, tal como se ha comentado antes, dependiendo del verbo en concreto, la representación formal del significado de la frase puede variar; en otras palabras, no hay una plantilla o modelo concreto que permita generar la representación del significado para cualquier verbo y esta depende el verbo en cuestión y cómo se ha formalizado el conocimiento.

En resumen, el analizador semántico precisa tanto de **conocimiento sobre el árbol sintáctico** como de **conocimiento sobre el ejemplo** concreto o dominio concreto de la frase que se analiza para poder crear de forma automática la representación del significado.

Dado que hay un número infinito de posibles árboles sintácticos para una gramática, diseñar un analizador semántico específico para cada árbol es totalmente inviable. Es por eso por lo que la estructura del proceso de análisis semántico dirigido por la sintaxis presentada en la figura 1, que separa el análisis sintáctico del análisis semántico, no es muy común en la realidad. Lo que se hace es integrar el conocimiento semántico en el propio proceso de generación del árbol sintáctico. Para

ello, se añade el conocimiento semántico a las reglas gramaticales y a las entradas léxicas que conforman la gramática, en lo que se conoce como ***rule-to-rule hypothesis*** (Bach, 1976).

Accede a los ejercicios de autoevaluación a través del aula virtual

7.3. Anotaciones semánticas a las gramáticas libres de contexto

Accede al vídeo «Anotaciones semánticas de las gramáticas independientes de contexto» a través del aula virtual

Una gramática libre de contexto se puede aumentar añadiéndole a las reglas anotaciones sobre el conocimiento semántico para así, poder implementar el análisis semántico guiado por la sintaxis.

Las **anotaciones semánticas** son instrucciones que especifican cómo generar la representación del significado de una construcción a partir de los significados de sus partes constituyentes.

Una regla con anotaciones semánticas tiene la siguiente estructura:

$$A \rightarrow \alpha_1 \ \ldots \ \alpha_n \quad \{ f\ (\alpha_j.sem, \ldots, \alpha_k.sem) \}$$

La anotación semántica que se añade a la notación básica de una regla libre de contexto se muestra en la parte derecha de los constituyentes sintácticos, dentro del símbolo de las llaves $\{\ldots\}$. Esta notación establece que la representación del significado asignada a la construcción A, llamada $A.sem$, se puede calcular ejecutando la función f en un subconjunto de las anotaciones semánticas de los

constituyentes de A. Donde $\alpha_1 \ldots \alpha_n$ son las partes constituyentes de A y sus anotaciones semánticas correspondientes son $\alpha_j.sem, \ldots, \alpha_k.sem$.

Esta notación es muy abstracta y, de hecho, para modelar las anotaciones semánticas es necesario utilizar algún lenguaje formal que permita representar el significado de una construcción y sus constituyentes. Es muy común utilizar lógica de primer orden y la notación de cálculo lambda para implementar las anotaciones semánticas y así, aumentar las reglas que componen la gramática.

Los **principios que se utilizan para implementar la hipótesis *rule-to-rule***, es decir, anotar semánticamente las reglas de una gramática, utilizando lógica de primer orden y la notación de cálculo lambda son:

1	Asociar expresiones del cálculo lambda complejas, similares a una función, a las reglas léxicas.
2	Copiar el valor semántico del único constituyente a la construcción cuando la regla gramatical solo tiene un constituyente.
3	Aplicar la semántica de uno de los constituyentes de una regla gramatical a la semántica de otro de los constituyentes de la regla como si fuera una función.

Figura 41. Principios para implementar la hipótesis *rule-to-rule*.

Para explicar estos tres principios se utiliza un ejemplo porque las anotaciones semánticas de las reglas de una gramática son altamente dependientes del conocimiento sobre el ejemplo concreto para el que se va a representar el significado.

Si se quiere analizar semánticamente la frase «Matías abrió un restaurante», es necesario aumentar una gramática libre de contexto con las correspondientes anotaciones semánticas. La gramática que define limitadamente el lenguaje natural del español y permite analizar la frase propuesta es la siguiente:

a) NP → `Matías`

b) N → `restaurante`

c) Det → `un`

d) V_t → `abrió`

1) O → `SN SV`

2) SN → `NP`

3) SN → `Det N`

4) SV → V_t `SN`

Donde:

- Las reglas de la *a)* a la *d)* son reglas léxicas, definen las categorías gramaticales y formarían parte del lexicón.
- Las reglas del 1 al 4 son reglas gramaticales y permiten definir la estructura sintagmática de la oración.

Primer principio sobre las anotaciones semánticas

Según el primer principio, las reglas léxicas se anotan con expresiones del cálculo lambda complejas. La regla *a)* hace referencia a que la palabra «Matías» es un nombre propio (*NP*). El significado de un nombre propio es el nombre en sí mismo. Entonces, en este caso, la anotación semántica más sencilla sería asignar a esta regla una constante, lo que representado en lógica de primer orden sería:

a) NP → `Matías` $\{Matías\}$

El mismo principio se puede aplicar a otras reglas léxicas donde las anotaciones son más complejas. Por ejemplo, la regla *b)* indica que la palabra «restaurante» es un nombre (N). A nivel conceptual, un restaurante es un tipo de objeto porque puede haber múltiples restaurantes, todos con una serie de características comunes, con sus particularidades e identificados por diferentes nombres. Por lo tanto, un restaurante en concreto será un objeto de la clase `Restaurante`, lo que se representa

en lógica de primer orden como $Restaurante(r)$, donde r es un elemento de la clase $Restaurante$.

Sin embargo, para representar un elemento no especificado de la clase se utiliza una variable r en la notación lambda (λr). Entonces la regla quedaría anotada de la siguiente forma:

$$\text{b) N} \rightarrow \texttt{restaurante}\ \{\lambda r.\, Restaurante(r)\}$$

Por consistencia con la anotación de la regla *b)*, la regla *a)* se podría rescribir de una forma más formal y utilizando variables de la siguiente forma:

$$\text{a) NP} \rightarrow \texttt{Matías}\ \{\ \lambda m.\, m(Matías)\ \}$$

Además, expresiones aún más complejas en lógica de primer orden y en la notación de cálculo lambda son necesarias para anotar la regla *c)* y la regla *d)*:

$$\text{c) Det} \rightarrow \texttt{un}\ \{\ \lambda P.\, \lambda Q.\, \exists x P(x)\ \Lambda\ Q(x)\ \}$$

$$\text{d) } V_t \rightarrow \texttt{abrió}\ \{\ \lambda w.\, \lambda z.\, w(\lambda x.\, \exists e\ Abierto(e)\ \Lambda\ PersonaQueAbre(e,z)\ \Lambda\ CosaAbierta(e,x))\ \}$$

Por ejemplo, en la regla *d)*, que representa el verbo transitivo (V_t) «abrió»; se utiliza un objeto, $Abierto(e)$, y dos relaciones $PersonaQueAbre(e,z)$ y $CosaAbierta(e,x)$ para modelar el hecho de abrir e, la persona z que realiza la acción de abrir y la cosa x que se abre.

Segundo principio sobre las anotaciones semánticas

El segundo principio indica que en las reglas gramaticales se copia el valor semántico del único constituyente a la construcción. Una regla de este tipo es la regla *2),* que indica que un único nombre propio (NP) puede constituir un sintagma nominal (SN). Por lo tanto, aplicando este principio se copia el valor semántico del único

constituyente a la construcción, es decir, que el significado del nombre propio ($NP.sem$) es el significado del sintagma nominal porque es el único elemento que lo compone.

Por ello, la regla anotada semánticamente sería la siguiente:

$$2)\ \text{SN} \rightarrow \texttt{NP}\ \{\ NP.sem\ \}$$

Entonces, al seguir las reglas *2)* y *a)*, si en el análisis semántico se detecta la palabra «Matías» y se sabe que es un nombre propio (NP) y que conforma un sintagma nominal (SN), por el principio de composición se va a poder inferir que el significado del sintagma nominal es el significado del nombre propio. Por ende, el sintagma nominal (SN) «Matías» tiene como significado el concepto `Matías`, lo que se representaría formalmente como:

$$SN.sem\ =\ NP.sem\ =\ \{\ \lambda m.m(Matías)\ \}$$

Tercer principio sobre las anotaciones semánticas

El tercer principio indica que, en las reglas gramaticales, se aplica la semántica de uno de los constituyentes de una regla a la semántica de otro de los constituyentes de la regla como si fuera una función. Por ejemplo, la regla *3)* indica que un sintagma nominal (SN) está formado por un determinante (`Det`) y un nombre (`N`). Entonces, el significado del sintagma nominal es el resultado de aplicar el significado del determinante ($Det.sem$) al significado del nombre (*N.sem*), lo que genera la siguiente regla anotada:

$$3)\ \text{SN} \rightarrow \texttt{Det}\ \texttt{N}\ \{\ Det.sem\ (N.sem)\ \}$$

A partir de las reglas *3)*, *b)* y *c),* y aplicando el principio de composición, es posible realizar la composición de los significados de la palabra «un» y la palabra

«restaurante» que forman el sintagma nominal «un restaurante» de la siguiente forma:

$$Det.sem \ = \ \{ \boldsymbol{\lambda P}.\lambda Q.\exists x \boldsymbol{P}(\boldsymbol{x}) \wedge Q(x) \}$$
$$N.sem \ = \ \{ \lambda r. \boldsymbol{Restaurante}(\boldsymbol{r}) \}$$
$$\{ Det.sem \ (N.sem) \} \ = \ \{ \lambda Q.\exists x \ Restaurante(x) \wedge Q(x) \}$$

Por lo tanto, el sintagma nominal (SN) «un restaurante» tendría como significado $\{ \lambda Q.\exists x \ Restaurante(x) \wedge Q(x) \}$ en la oración a analizar, lo que se puede interpretar como que existe (∃) un elemento x del tipo `Restaurante`.

El tercer principio se aplica de la misma forma a las reglas *1)* y *4),* dando como resultado las siguientes reglas anotadas:

$$1) \ O \rightarrow SN \ \ SV \ \{ SN.sem \ (SV.sem) \}$$
$$4) \ SV \rightarrow V_t \ \ SN \ \{ Vt.sem \ (SN.sem) \}$$

A partir de las reglas *4)* y *d)* y la composición del significado del sintagma nominal (SN) «un restaurante» se puede obtener el significado del sintagma verbal (SV) «abrió un restaurante» de la siguiente forma:

$$Vt.sem \ = \ \{ \lambda w.\lambda z.w(\boldsymbol{\lambda x}.\exists \boldsymbol{e} \ \boldsymbol{Abierto}(\boldsymbol{e}) \wedge \boldsymbol{PersonaQueAbre}(\boldsymbol{e},\boldsymbol{z}) \wedge \boldsymbol{CosaAbierta}(\boldsymbol{e},\boldsymbol{x})) \}$$
$$SN.sem \ = \ \{ Det.sem \ (N.sem) \} \ = \ \{ \boldsymbol{\lambda Q}.\exists x \ Restaurante(x) \wedge \boldsymbol{Q}(\boldsymbol{x}) \}$$
$$\{ Vt.sem \ (SN.sem) \} \ = \ \{ \lambda z.(\exists x \ Restaurante(x) \wedge \exists e \ Abierto(e) \wedge PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)) \}$$

El sintagma verbal (SV) «abrió un restaurante» tiene como significado la expresión:

$$\{ \lambda z.(\exists x \ Restaurante(x) \wedge \exists e \ Abierto(e) \wedge PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)) \}$$

Esto se puede interpretar como que existe (∃) un elemento e del tipo `Abierto` que se relaciona a través de la relación $PersonaQueAbre$ con un objeto (desconocido) z y que también se relaciona a través de la relación $CosaAbierta$ con un elemento x del tipo $Restaurante$.

Para finalizar, a partir de la regla *1)*, del significado del sintagma nominal (SN) «Matías» y del significado del sintagma verbal (SV) «abrió un restaurante», computados previamente, se puede obtener por el principio de composición el significado de la oración «Matías abrió un restaurante» de la siguiente forma:

$$SN.sem = NP.sem = \{ \boldsymbol{\lambda m}.\boldsymbol{m}(\boldsymbol{Matías}) \}$$
$$SV.sem = \{ Vt.sem\ (SN.sem) \} = \{ \boldsymbol{\lambda z}.(\exists x\ Restaurante(x)\ \Lambda\ \exists e\ Abierto(e)\ \Lambda\ PersonaQueAbre(e, \boldsymbol{z})\ \Lambda\ CosaAbierta(e,x)) \}$$
$$O.sem = \{ SN.sem\ (SV.sem) \} = \{ \exists x\ Restaurante(x)\ \Lambda\ \exists e\ Abierto(e)\ \Lambda\ PersonaQueAbre(e, Matías)\ \Lambda\ CosaAbierta(e,x) \}$$

Entonces la oración (O) «Matías abrió un restaurante» tiene como significado la expresión:

$$\{ \exists x\ Restaurante(x)\ \Lambda\ \exists e\ Abierto(e)\ \Lambda\ PersonaQueAbre(e, Matías)\ \Lambda\ CosaAbierta(e,x) \}$$

Esta representación del significado de la frase se puede interpretar como que existe (∃) un elemento e del tipo $Abierto$ que se relaciona a través de la relación $PersonaQueAbre$ con un objeto llamado $Matías$ y que también se relaciona a través de la relación $CosaAbierta$ con un elemento x del tipo $Restaurante$.

En resumen, la gramática libre de contexto anotada semánticamente, útil para que la información sintáctica representada en forma de reglas se complemente con información semántica y que permite realizar el análisis semántico de la oración del ejemplo anterior sería la siguiente:

a) NP → `Matías` $\{ \lambda m.m(Matías) \}$

b) N → `restaurante` $\{ \lambda r.Restaurante(r) \}$

c) Det → `un` $\{ \lambda P.\lambda Q.\exists x P(x) \Lambda Q(x) \}$

d) V_t → `abrió` $\{ \lambda w.\lambda z.w(\lambda x.\exists e\ Abierto(e) \Lambda$

$PersonaQueAbre(e,z) \Lambda CosaAbierta(e,x) \}$

1) O → `SN SV` $\{ SN.sem\ (SV.sem) \}$

2) SN → `NP` $\{ NP.sem \}$

3) SN → `Det N` $\{ Det.sem\ (N.sem) \}$

4) SV → V_t `SN` $\{ Vt.sem\ (SN.sem) \}$

Accede a los ejercicios de autoevaluación a través del aula virtual

7.4. Análisis semántico integrado con el análisis sintáctico

Accede al vídeo «Análisis semántico integrado con el análisis sintáctico» a través del aula virtual

El análisis semántico se puede realizar en paralelo con el análisis sintáctico utilizando una gramática que incorpore la semántica junto con la representación sintáctica.

Las gramáticas libres de contexto, presentadas en la sección anterior, permiten realizar el análisis sintáctico y semántico simultáneamente. Esto marca una diferencia con la técnica explicada al inicio de este tema, que requiere del resultado del análisis sintáctico como entrada para poder realizar el análisis semántico.

La implementación del análisis semántico integrado con el sintáctico es posible gracias a que la representación de significado para una construcción se puede obtener tan pronto como todas sus partes constituyentes sean procesadas.

Entonces, la **principal ventaja** de utilizar este enfoque integrado radica en que si se encuentra una construcción semántica que no tiene sentido cuando se está creando la representación de significado, se puede considerar que el árbol sintáctico correspondiente tampoco va a ser válido. Por tanto, las consideraciones semánticas que se tienen en cuenta durante el análisis sintáctico permiten simplificar este tipo de análisis.

Sin embargo, este hecho también conlleva asociado una de las **principales desventajas** de integrar la semántica directamente en el analizador sintáctico. Puede ser que se dedique un esfuerzo considerable al análisis semántico de componentes de la oración que al final no contribuyan a desarrollar un árbol sintáctico consistente. De esta forma, puede ser que se hayan realizado operaciones en el análisis semántico que son totalmente innecesarias para un análisis sintáctico exitoso.

Por desgracia no existe una respuesta a la cuestión de si la ganancia obtenida de incorporar la semántica ya desde el inicio en el análisis sintáctico compensa el coste de realizar algún procesamiento semántico innecesario. Por lo que no se puede afirmar que haya un enfoque mejor que otro para realizar el análisis semántico.

Accede a los ejercicios de autoevaluación a través del aula virtual

7.5. Referencias bibliográficas

Bach, E. (1976). An extension of classical transformational grammar. In Problems of Linguistic Metatheory (Proceedings of the 1976 Conference). Michigan State University.

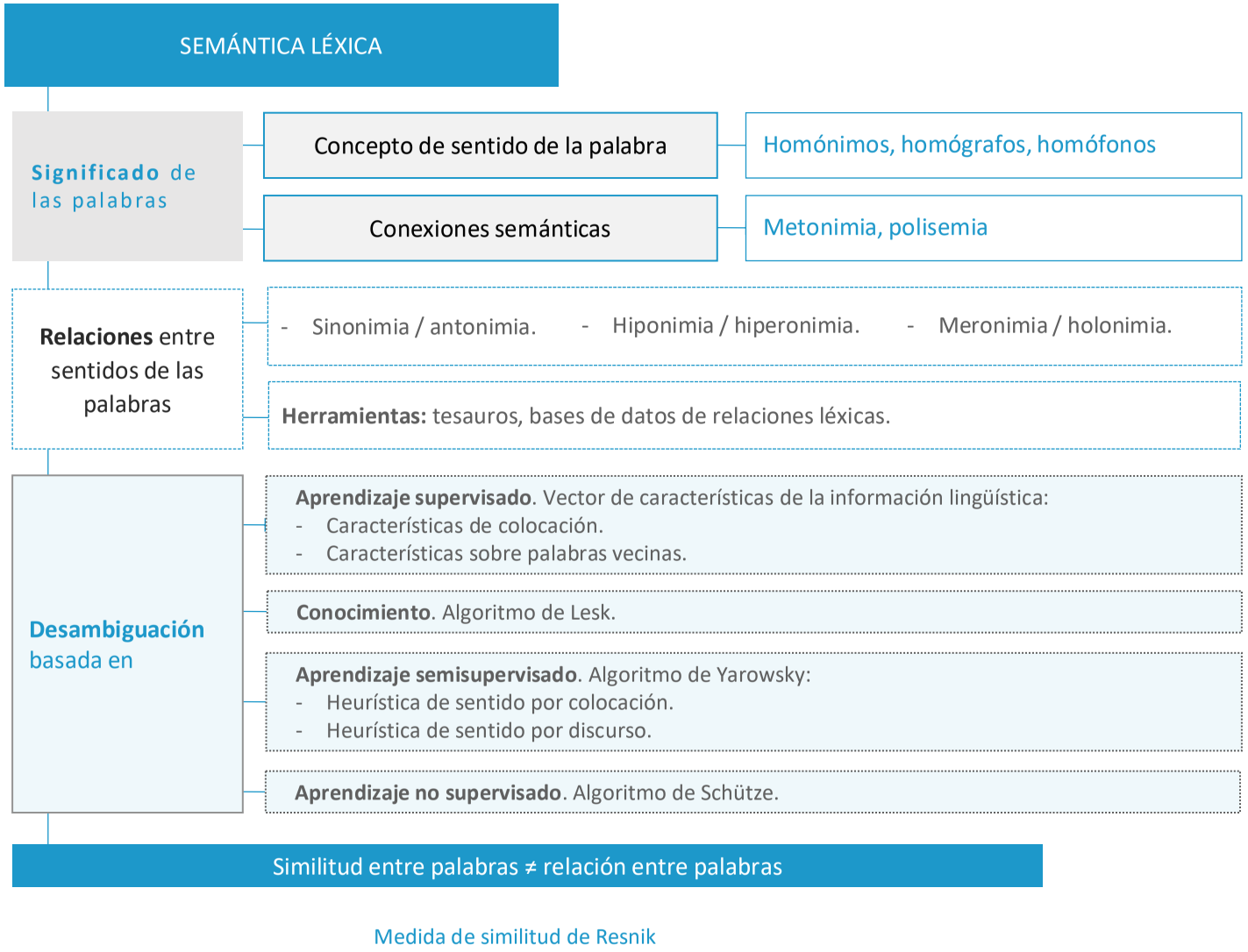
Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics*. New Jersey, Estados Unidos: Prentice-Hall.

Accede al vídeo «Resumen» a través del aula virtual

Tema 8. Semántica léxica

Índice

Esquema



Ideas clave

8.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Los apartados de los que consta este tema son:

- Significado de las palabras.
- Relaciones entre sentidos de las palabras.
- Desambiguación del sentido de las palabras.
- Similitud entre palabras.

Al finalizar el estudio de este tema serás capaz de:

- Definir los conceptos relevantes en al ámbito de la semántica léxica, campo de la lingüística que estudia el significado de las palabras.
- Identificar los diferentes tipos de relaciones entre los sentidos y significados de las palabras: homonimia, metonimia, sinonimia, antonimia, hiponimia…
- Desambiguar el sentido de las palabras aplicando técnicas de aprendizaje automático, ya sean de aprendizaje supervisado, no supervisado o semisupervisado.
- Calcular la similitud entre palabras utilizando tesauros.

Accede a los ejercicios de autoevaluación a través del aula virtual

8.2. Significado de las palabras

Accede al vídeo «Significado de las palabras» a través del aula virtual

La Real Academia Española (RAE, s. f.a) en su diccionario de la lengua española define **semántica léxica** como: «1. f. Rama de la semántica que estudia el significado de las palabras, así como las diversas relaciones de sentido que se establecen entre ellas».

La forma más sencilla de representar el significado de una palabra sería utilizando el **lema**. Tal como se vio en temas anteriores, se llama lema a la raíz de la palabra y, por lo tanto, es la forma gramatical de la palabra que se utiliza en los diccionarios. Por ejemplo, para la palabra *bancos* el lema es la forma «banco».

Un lema puede tener varios significados: en el caso del lema «banco» puede referirse al tipo de asiento donde pueden sentarse varias personas o a la empresa que realiza operaciones financieras, entre otras definiciones que aparecen en el diccionario de la RAE. A cada uno de los significados del lema «banco» se le llama un sentido de la palabra o simplemente sentido.

Un **sentido** es una representación de uno de los aspectos del significado de una palabra.

Cada uno de los sentidos de una palabra se puede representar añadiendo un superíndice a la forma ortográfica del lema. Entonces, los dos sentidos de la palabra *banco* se representarían como «banco^{1}» y «banco^{2}», respectivamente. Los diferentes sentidos de una palabra (en el ejemplo, el banco como un tipo de asiento («banco^{1}») y como una entidad financiera («banco^{2}»)) normalmente no tienen ninguna relación a nivel de significado entre sí, por eso se dice que «banco^{1}» y «banco^{2}» son homónimos, término que se aplica a cosas que tienen el mismo nombre.

Si una palabra tiene varios sentidos que no están relacionados entre sí, se dice que los sentidos son **homónimos** y a la relación entre dos de estos sentidos se le llama **relación de homonimia**.

Además, en el caso del *banco* como un tipo de asiento («banco[1]») y como una entidad financiera («banco[2]») se escriben de la misma forma y, por lo tanto, se dice que estos dos usos son **homógrafos**: según la RAE (s. f.b), una palabra «que tiene la misma grafía que otra».

Existe también otra forma en que dos palabras pueden ser homónimas, sería el caso de palabras que se escriben de forma diferente, pero que se pronuncian igual. Por ejemplo, la palabra *vello* (en referencia al pelo del cuerpo) y la palabra *bello* (hermoso) son **homónimas**: según la RAE (s. f.c), una palabra «que se pronuncia como otra, pero tiene diferente origen o significado muy distante».

Además, *vello* (pelo del cuerpo) y *bello* (hermoso) son dos lemas que suenan igual por lo que se dice que son **homófonos**: según la RAE (s. f.d), una palabra «que suena igual que otra, pero que tiene distinto significado y puede tener distinta grafía».

- La homofonía es una de las causas de que se cometan errores ortográficos en la vida real y que también afecta el procesamiento del lenguaje natural cuando se realiza el reconocimiento automático del habla.
- De forma similar, la homografía afecta a la tarea de conversión de texto al habla.
- La homonimia es la principal causa de diferentes errores que se dan en el procesamiento del lenguaje natural. Para tratarla se aplican técnicas de desambiguación del sentido de la palabra.

La **desambiguación del sentido de la palabra** es la tarea de determinar qué sentido de una palabra se usa en un contexto particular.

A diferencia de lo que se acaba de exponer, a veces existe una **conexión semántica** entre los sentidos de las palabras. Volviendo al ejemplo de la palabra *banco*, puede

existir un tercer significado que sería la oficina donde opera la entidad financiera («banco^{3}»). Este último significado sí está relacionado con «banco^{2}» (la entidad financiera); estos dos sentidos, «banco^{2}» y «banco^{3}», tienen una conexión semántica. Sin embargo, «banco^{3}» (la oficina de la entidad financiera) sigue sin tener ninguna relación semántica con «banco^{1}» (el tipo de asiento).

Por otro lado, la oficina bancaria («banco^{3}») coge el nombre de la propia organización, es decir, la entidad financiera («banco^{2}»). Lo mismo pasa con otros ejemplos como la palabra *universidad,* que hace referencia a la institución de enseñanza superior, «universidad^{1}», pero también a los edificios de las cátedras y oficinas de la institución universitaria, «universidad^{2}». Este tipo de relación semántica entre los sentidos de las palabras se llama metonimia.

La **metonimia** es una forma de emplear una palabra en un sentido distinto al que propiamente le corresponde, pero con el que tiene algún tipo de conexión.

Concretamente, la metonimia «consiste en designar algo con el nombre de otra cosa tomando el efecto por la causa o viceversa, el autor por sus obras, el signo por la cosa significada, etc.; p. ej., las canas por la vejez; leer a Virgilio, por leer las obras de Virgilio» (RAE, s. f.e). Entonces, el sentido «banco^{3}» (la oficina de la entidad financiera) tiene una **relación de metonimia** con «banco^{2}» (la entidad financiera), así como «universidad^{1}» (institución de enseñanza superior) con la «universidad^{2}» (los edificios de las cátedras y oficinas de la institución universitaria).

La relación semántica de la metonimia es un tipo particular de polisemia, el nombre genérico que se utiliza para definir cualquier relación semántica entre dos significados, ya que la palabra polisemia se refiere a la pluralidad de significados.

Si una palabra tiene varios sentidos que están relacionados entre sí, se dice que los sentidos son **polisémicos** y a la relación entre dos de estos sentidos se le llama **relación de polisemia**.

Los diccionarios son los repertorios donde se recogen, según un orden determinado, las palabras de una lengua acompañadas de su definición o explicación. Incluyen una descripción detallada y comprensible para un humano de los diferentes sentidos de las palabras. En el procesamiento del lenguaje natural, el formato que siguen los diccionarios tradicionales no es el más útil para obtener y entender los significados de las palabras.

PARA EL PROCESAMIENTO AUTOMÁTICO ES NECESARIO

1. Definir el sentido de una palabra a través de su relación con los sentidos de otras palabras.
2. Agrupar los sentidos de las palabras.

Figura 42. Requisitos a la hora de elaborar un repertorio de los sentidos de las palabras en PLN.

Por ejemplo, para el procesamiento del lenguaje natural puede ser útil definir que «rojo» y «azul» son lemas del mismo tipo, concretamente del tipo `color`. Además de establecer que pertenecen al mismo grupo y son complementarios, lo que indica que algo que es rojo no puede ser azul. También puede ser interesante definir que la sangre es roja y el mar es azul. Si se tiene una base de datos suficientemente exhaustiva con estas relaciones entre los sentidos de las palabras será posible realizar tareas semánticas muy sofisticadas.

Accede a los ejercicios de autoevaluación a través del aula virtual

8.3. Relaciones entre sentidos de las palabras

Accede al vídeo «Relaciones entre los sentidos de las palabras» a través del aula virtual

Las relaciones más comunes que se dan entre los sentidos de las palabras son la sinonimia y la antonimia. También destacan la hiponimia y la hiperonimia, y, aunque sea menos común, la meronimia.

Cuando los sentidos de dos palabras diferentes y, por extensión, los significados de sus dos lemas son idénticos o casi idénticos, se puede decir que hay una relación de **sinonimia** entre ambos. Por ejemplo, las palabras *empezar* y *comenzar* son sinónimos ya que, si se intercambian en una frase, esta mantiene prácticamente el mismo significado.

Una palabra es un **sinónimo** de otra si tiene el mismo significado o un significado muy parecido.

La **antonimia** es la relación contraria. Los antónimos son palabras que tienen significados opuestos como, por ejemplo, *claro* y *oscuro* o *antes* y *después*. Dos sentidos pueden ser antónimos si definen una oposición binaria o están en extremos opuestos de alguna escala. Este es el caso de *largo* y *corto*, *rápido* y *lento* o *grande* y *pequeño*, porque se encuentran en extremos opuestos de la escala de longitud, de tiempo y de tamaño, respectivamente. Además, se puede definir un grupo de antónimos llamados **reversos,** que describen cambios o movimientos en direcciones opuestas como *abierto* y *cerrado* o *subir* y *bajar*.

Una palabra es un **antónimo** de otra si expresa una idea opuesta o contraria.

Un sentido es un hipónimo de otro si el primer sentido es más específico que el segundo, es decir, si el primero es una subclase del segundo. Por ejemplo, *coche* es hipónimo de *vehículo; perro* lo es a *animal* y *mango* lo es a *fruta* porque el coche es un tipo de vehículo, el perro un tipo de animal y el mango un tipo de fruta.

Una palabra es un **hipónimo** de otra si su significado incluye el significado de la otra palabra.

La relación contraria a la hiponimia es la **hiperonimia**. Entonces, se dice que *vehículo* es hiperónimo de *coche*, *animal* es hiperónimo de *perro* y *fruta* es hiperónimo de *mango* porque la clase que representa los vehículos incluye como miembros a todos

los coches, la que representa a los animales incluye a los perros y la que representa a las frutas incluya a los mangos.

Una palabra es un **hiperónimo** de otra si su significado está incluido en el significado de la otra palabra.

Formalmente se define que un sentido A es hipónimo de un sentido B si todo lo que es A también es B, entonces ser un A implica ser un B, lo que en lógica de primer orden se escribiría como:

$$\forall x\, A(x) \Rightarrow B(x)$$

La hiponimia suele ser una **relación transitiva**; si A es hipónimo de B y B es un hipónimo de C, entonces A es un hipónimo de C.

En las ontologías, representaciones semánticas basadas en lógica descriptiva, se modelan las relaciones de hiponimia e hiperonimia a través de la **jerarquía IS-A**. La expresión A IS-A B indica que un sentido *A* es hipónimo de un sentido *B*, o que un sentido *B* es hiperónimo de un sentido *A*.

Una última relación destacada es la **meronimia** que define una relación del tipo parte-todo entre los sentidos de las palabras. Por ejemplo, una rueda es una parte de un coche, por lo que se dice que *rueda* es merónimo de *coche* o, al contario, que *coche* es **holónimo** de *rueda*.

Una palabra es un **merónimo** de otra si su significado mantiene con el significado de la otra palabra una relación de la parte respecto al todo.

Para poder realizar tareas de procesamiento del lenguaje natural es necesario recoger las relaciones entre los sentidos de las palabras de un diccionario. De hecho, en literatura se utiliza el término **tesauro**, *thesaurus*, *thesauri* o tesoro para referirse a los diccionarios que contienen una lista de palabras con sus sinónimos y sus

antónimos. En lingüística, los términos que conforman un tesauro se relacionan entre sí para mostrar las relaciones entre significados.

Las llamadas **bases de datos de relaciones léxicas** contienen un conjunto de lemas, cada uno de los cuales está anotado con el posible conjunto de sentidos de la palabra. Cada uno de los sentidos contiene, además de su definición en un formato tipo glosario y una lista de sinónimos. No es obligatorio que las bases de datos de relaciones léxicas contengan la pronunciación los lemas, sin embargo, es imprescindible que estas contengan detalles de las relaciones entre lemas.

La figura 1 muestra un ejemplo de algunas relaciones entre sentidos en WordNet (Fellbaum, 1998), la que es la principal base de datos léxica para procesamiento del lenguaje natural en inglés.

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	*breakfast*^{1} → *meal*^{1}
Hyponym	Subordinate	From concepts to subtypes	*meal*^{1} → *lunch*^{1}
Instance Hypernym	Instance	From instances to their concepts	*Austen*^{1} → *author*^{1}
Instance Hyponym	Has-Instance	From concepts to concept instances	*composer*^{1} → *Bach*^{1}
Member Meronym	Has-Member	From groups to their members	*faculty*^{2} → *professor*^{1}
Member Holonym	Member-Of	From members to their groups	*copilot*^{1} → *crew*^{1}
Part Meronym	Has-Part	From wholes to parts	*table*^{2} → *leg*^{3}
Part Holonym	Part-Of	From parts to wholes	*course*^{7} → *meal*^{1}
Substance Meronym		From substances to their subparts	*water*^{1} → *oxygen*^{1}
Substance Holonym		From parts of substances to wholes	*gin*^{1} → *martini*^{1}
Antonym		Semantic opposition between lemmas	*leader*^{1} ⟺ *follower*^{1}
Derivationally Related Form		Lemmas w/same morphological root	*destruction*^{1} ⟺ *destroy*^{1}

Figura 43. Relaciones entre sentidos en WordNet.
Fuente: Jurafsky y Martin, 2009.

Accede a los ejercicios de autoevaluación a través del aula virtual

8.4. Desambiguación del sentido de las palabras

Accede al vídeo «Desambiguación del sentido de las palabras (parte 1)», «Desambiguación basada en aprendizaje supervisado (parte 2)», «Desambiguación basada en conocimiento. Algoritmo de *Lesk* simplificado (parte 3), «Desambiguación basada en conocimiento: algoritmo de Lesk original (parte 4)» a través del aula virtual

Los métodos para el análisis semántico composicional que se han estudiado en el tema anterior no tienen en cuenta que una palabra puede tener más de un significado. De hecho, esos métodos obvian la ambigüedad léxica que es una realidad existente y es un problema importante en el procesamiento del lenguaje natural.

La **desambiguación del sentido de las palabras** es la tarea de seleccionar el sentido correcto para una palabra. Los algoritmos de desambiguación del sentido toman como entrada una palabra en su contexto y una lista de posibles significados de esa palabra y devuelven como salida el sentido correcto para ese uso concreto de la palabra.

Existen diferentes opciones para implementar un algoritmo de desambiguación del sentido de las palabras. La primera opción se basa en aplicar técnicas de **aprendizaje automático supervisado** para aprender un modelo clasificador que permita desambiguar las palabras. Estos algoritmos de desambiguación basados en aprendizaje supervisado requieren tener un corpus de palabras etiquetadas con sus sentidos correctos para poder entrenar el clasificador. Tener acceso a este tipo de datos etiquetados puede ser complejo y es por eso por lo que, aunque los algoritmos de desambiguación basados en aprendizaje supervisado sean los que proporcionan mejores resultados, a veces no se utilizan por el elevado coste asociado a la obtención de los datos etiquetados.

Si no se dispone de un corpus etiquetado, una alternativa es utilizar diccionarios, tesauros u otras bases de conocimiento para realizar un entrenamiento indirecto, aplicando algoritmos de aprendizaje supervisado débil. Este tipo de métodos de desambiguación se llaman **algoritmos de desambiguación basados en conocimiento** y no utilizan textos que hayan sido etiquetados manualmente, sino que utilizan grandes bases de conocimiento.

Otra opción a los algoritmos de desambiguación basados en aprendizaje supervisado y a los basados en conocimiento es aplicar técnicas de **aprendizaje semisupervisado** o ***bootstrapping***. Estos algoritmos de desambiguación basados en aprendizaje semisupervisado no requieren de grandes recursos lingüísticos generados a mano, es decir, ni de un gran conjunto de datos de entrenamiento ni de un gran diccionario, sino que para funcionar, es suficiente con tener un pequeño conjunto de datos de entrenamiento etiquetados a mano.

Una última alternativa, que pretende evitar la tarea compleja y costosa de construir un corpus de palabras etiquetadas con los sentidos para la desambiguación de las palabras, se basa en técnicas de **aprendizaje no supervisado**.

Los algoritmos basados en este tipo de aprendizaje realizan la desambiguación sin utilizar definiciones de los sentidos de las palabras por humanos. El conjunto de sentidos de cada palabra se crea automáticamente a partir de las instancias de la palabra disponibles en los datos de entrenamiento. Esta tarea se llama también **inducción del sentido de las palabras**.

Desambiguación basada en aprendizaje supervisado

Los algoritmos de desambiguación basados en el aprendizaje automático supervisado utilizan un conjunto de instancias etiquetadas para entrenar un clasificador. Una vez entrenado, este sirve para predecir el mejor sentido de las palabras ambiguas. Por lo tanto, el resultado del entrenamiento es un modelo clasificador capaz de asignar

etiquetas de sentido a las palabras no etiquetadas que aparecen en un contexto determinado.

En una primera opción de implementación de los algoritmos de aprendizaje supervisado para aprender desambiguaciones, se parte de un conjunto de palabras ambiguas y se empieza preseleccionando un subconjunto de sentidos posibles para estas. Así, para cada palabra, se selecciona un número de instancias de la aparición de esa palabra en un corpus. Estas instancias se etiquetan a mano con el sentido correcto, según el contexto en el que aparecen en el corpus. Una vez se tienen todos los ejemplos etiquetados, estos se utilizan para entrenar el clasificador.

Como en este caso se trabaja con un conjunto reducido de palabras y el conjunto reducido de sentidos, se dice que el algoritmo de aprendizaje supervisado se entrena con una **muestra léxica**, con lo que aprende un modelo clasificador que permite desambiguar una única palabra o, como máximo, un conjunto reducido de palabras concretas, algo que puede ser poco práctico.

En una segunda opción de implementación de los algoritmos de aprendizaje supervisado, se utiliza un corpus de palabras ya etiquetadas con su sentido para aprender la desambiguación de un texto entero. Es decir, que se aprende a desambiguar todas las palabras del texto y no solo algunas palabras concretas, como era el caso de utilizar una muestra léxica.

El algoritmo de desambiguación que utiliza un **lexicón** para entrenar requiere de un conjunto muy grande de etiquetas porque cada lema tiene su propio set de etiquetas. Gestionar este conjunto enorme puede representar un problema y reducir la aplicabilidad de estas técnicas de aprendizaje supervisado basadas en un corpus etiquetado.

Los algoritmos de desambiguación basados en el aprendizaje supervisado, ya sean los que para entrenar utilizan una muestra léxica etiquetada a mano o los que utilizan

un corpus ya etiquetado, tienen que extraer características de texto y, a partir de ellas, asignar etiquetas con el sentido correcto.

Entonces, el primer paso en los algoritmos de aprendizaje supervisado es **extraer características** que sean predictivas de los sentidos de las palabras. Los algoritmos modernos de desambiguación del sentido extraen información del contexto que rodea la palabra a desambiguar (Weaver, 1955). Para ello se utiliza una ventana que incluye la palabra ambigua y las palabras anteriores y posteriores, también llamadas palabras de contexto, y se extrae un **vector de características de la información lingüística** de las palabras de contexto contenidas en la ventana. Dicho vector consta de valores numéricos o nominales y contiene dos tipos de características:

- Características de colocación.
- Características sobre las palabras vecinas.

Características de colocación

Codifican información sobre la posición de las palabras de contexto y su relación con la palabra ambigua. Algunas características típicas que se pueden extraer de las palabras de contexto incluyen la palabra en sí, la raíz de la palabra y su categoría gramatical o morfosintáctica. Así, estas características codifican información léxica y gramatical que a menudo puede ayudar a identificar con precisión el sentido de la palabra.

Por ejemplo, un vector de características de colocación extraído de una ventana con dos palabras a la derecha y a la izquierda de la palabra objetivo, donde las características son las palabras, sus categorías gramaticales y los pares de palabras, tendría la siguiente forma:

$$[\, w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}, {w_{i-2}}^{i-1}, {w_i}^{i+1} \,]$$

Donde:

- w_{i-2} es la palabra dos posiciones a la izquierda de la palabra ambigua.
- POS_{i-2} es la categoría gramatical de la palabra w_{i-2}.
- ${w_{i-2}}^{i-1}$ es el par de palabras w_{i-2} y w_{i-1}.

Ejemplo ilustrativo 1. Vector de características de colocación para desambiguar la palabra *bass* (bajo) en la oración «*An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps*» del corpus WSJ.

Se define una ventana con dos palabras a la derecha y a la izquierda de la palabra *bass*:

guitar	***and***	***bass***	***player***	***stand***
w_{i-2}	w_{i-1}		w_{i+1}	w_{i+2}

Tabla 7. Ventana con la palabra *bass* como palabra objetivo.

Para calcular algunas de las características se realiza el análisis morfosintáctico y se identifican las siguientes categorías gramaticales de las palabras de contexto:

guitar	***and***	***bass***	***player***	***stand***
NN	CC		NN	VB

Tabla 8. Categorías gramaticales de las palabras contexto.

Como características se utilizan las propias palabras de contexto, sus categorías gramaticales y los pares de palabras posicionadas antes y después de la palabra *bass*. El vector de características de colocación sería:

```
[guitar, NN, and, CC, player, NN, stand, VB, and
             guitar, player stand]
```

Características sobre las palabras vecinas

El segundo tipo de características codifican información sobre la llamada *bag-of-words* (bolsa de palabras), es decir, sobre un conjunto de palabras sin considerar su

orden y que rodean a la palabra ambigua. Las características que se extraen de una bolsa de palabras son eficaces para capturar el **tema general del discurso** en el que ocurre la palabra ambigua. Además, el tema del discurso tiende a identificar a su vez los sentidos de las palabras que son específicas en un cierto contexto.

En el enfoque más simple de bolsa de palabras, el contexto de una palabra ambigua se representa como un vector de características donde cada característica binaria indica si una palabra de un vocabulario aparece o no en el contexto. El vocabulario lo conforman un subconjunto de palabras útiles o de uso frecuente y preseleccionadas del corpus de entrenamiento. La región de contexto se define como una ventana de tamaño fijo donde la palabra ambigua se encuentra en el centro y que contiene un número generalmente pequeño de palabras situadas a la izquierda y a la derecha de la palabra ambigua.

> **Ejemplo ilustrativo 2.** Vector de características sobre las palabras vecinas para desambiguar la palabra *bass* (bajo) en la oración «*An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps*» del corpus WSJ.
>
> Se define la siguiente bag-of-words (bolsa de palabras) con las 12 palabras que aparecen más frecuentemente en frases donde se encuentra la palabra *bass* en el corpus WSJ:
>
> `[fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band]`
>
> Se utiliza una ventana de **tamaño nueve**: ventana que incluye cuatro palabras a la izquierda y cuatro a la derecha de la palabra ambigua. Por lo tanto, las palabras que quedarían dentro de la ventana son:
>
> *An electric guitar and bass player stand off to*
>
> En la ventana aparecen las palabras *guitar* y *player* y no aparecen ninguna de las otras palabras que forman parte de la bolsa (*bag-*

of-words). Entonces el vector de características sobre las palabras vecinas sería:

```
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]
```

Cualquier conjunto de datos de entrenamiento puede servir para extraer tanto características de colocación como características sobre las palabras vecinas y entrenar un clasificador de sentidos de las palabras con un algoritmo de aprendizaje automático supervisado. Sin embargo, el buen funcionamiento del clasificador va a depender de la cantidad de datos de entrenamiento con los que se cuente, es decir, de la disponibilidad de datos etiquetados con los sentidos. Debido a este problema, algunos investigadores han empezado a usar la Wikipedia como fuente de datos de entrenamiento (Mihalcea, 2007) (Ponzetto y Navigli, 2010).

Desambiguación basada en conocimiento

Los algoritmos de desambiguación que utilizan bases de conocimiento, ya sean diccionarios o tesauros, para realizar un entrenamiento indirecto aplican algoritmos de aprendizaje supervisado débil.

- En el caso de utilizar diccionarios, se aplica el algoritmo de Lesk, que se presenta a continuación.
- En el caso de utilizar un tesauro, se aplican métodos basados en gráficos (Agirre, López de Lacalle y Soroa, 2014) (Navigli y Lapata, 2010).

El llamado **algoritmo de Lesk** se refiere en realidad una familia de algoritmos que seleccionan el sentido de la palabra para el cual su definición en el diccionario comparte la mayor cantidad de palabras con los vecinos de la palabra ambigua.

La versión más simple del algoritmo de Lesk llamada **algoritmo de Lesk Simplificado** (Kilgarriff y Rosenzweig, 2000) compara la firma de la palabra ambigua *(signature),* que se corresponde con su definición en el diccionario, con las palabras de contexto

(context), es decir, con las palabras vecinas a la palabra ambigua. El pseudocódigo se presenta en la figura 3, donde la función COMPUTEOVERLAP devuelve el número de palabras que tienen en común los conjuntos de palabras *signature* y *context*

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

 best-sense ← most frequent sense for word
 max-overlap ← 0
 context ← set of words in sentence
 for each sense in senses of word do
  signature ← set of words in the gloss and examples of sense
  overlap ← COMPUTEOVERLAP(signature, context)
  if overlap > max-overlap then
      max-overlap ← overlap
      best-sense ← sense
 end
 return(best-sense)
```

Figura 44. Pseudocódigo del algoritmo de Lesk Simplificado.
Fuente: Jurafsky y Martin, 2009.

Ejemplo ilustrativo 3. Funcionamiento del algoritmo de Lesk Simplificado para desambiguar la palabra *bank* (banco) en la oración «*The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities*».

Del diccionario WordNet se obtienen las siguientes definiciones para la palabra *bank*:

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

Figura 45. Definiciones de la palabra *bank* en WordNet.

Se observa que en la definición y los ejemplos del sentido «bank[1]» aparecen dos palabras, *deposits* y *mortgage,* que también aparecen en el contexto de la palabra a desambiguar, es decir, en la frase bajo análisis.

Por el contrario, ni en la definición ni en los ejemplos del sentido «bank[2]» aparecen palabras relevantes que también aparezcan

en el contexto de la palabra a desambiguar. Es importante notar que los determinantes (por ejemplo, *«the»*), preposiciones, conjunciones y pronombres no se tienen en cuenta por no aportar información relevante.

Por lo tanto, el algoritmo de Lesk Simplificado escogería el sentido «bank[1]», que hace referencia a la entidad financiera, como sentido correcto para la palabra *bank* en la frase que se ha analizado.

La **versión original del algoritmo de Lesk** (Lesk, 1986) utiliza para el entrenamiento una experiencia incluso más indirecta que la versión simple del algoritmo. De hecho, la versión original lo que hace es comparar la firma de la palabra ambigua con las firmas de cada una de las palabras del contexto.

El principal problema de la versión original y la versión simplificada del algoritmo de Lesk es que las entradas del diccionario para las palabras ambiguas son cortas y puede que no se dé la oportunidad de que se solapen con las palabras del contexto (Lesk, 1986).

La solución es añadir todas las palabras en las oraciones del corpus etiquetado para un sentido de la palabra al cómputo de la firma de ese sentido. Esta versión del algoritmo se llama **Corpus Lesk** y es la que proporciona mejores resultados (Kilgarriff y Rosenzweig, 2000) (Vasilescu, Langlais y Lapalme, 2004). Además, el algoritmo de Corpus Lesk lo que hace es aplicar un peso a cada palabra coincidente en lugar de contar el número de palabras. Aplicar un peso sirve para dar menos relevancia a las palabras poco importantes, pero que se repiten mucho como, por ejemplo, los determinantes, preposiciones, conjunciones y pronombres.

Para calcular los pesos en el algoritmo de Corpus Lesk se utiliza una medida llamada ***inverse document frequency*** **(IDF)** y que se define para la palabra *i* según la siguiente fórmula:

$$idf_i = \log \frac{Ndoc}{nd_i}$$

Donde:

- $Ndoc$ es el número total de «documentos», es decir, definiciones y ejemplos.
- nd_i es el número de estos documentos que contienen la palabra *i*.

Se puede combinar el algoritmo de Lesk y los enfoques de aprendizaje automático supervisado añadiendo nuevas características similares a las computadas con el algoritmo de Lesk a la bolsa de palabras. Por ejemplo, las definiciones y las frases de ejemplo para el sentido de una palabra del diccionario se podrían usar en la bolsa de palabras utilizada en el aprendizaje supervisado (Yuret, 2004).

Desambiguación basada en aprendizaje semisupervisado

Los algoritmos de desambiguación basados en aprendizaje semisupervisado o *bootstrapping* no requieren de grandes recursos lingüísticos generados a mano, sino que para funcionar solo necesitan un pequeño conjunto de datos de entrenamiento etiquetados a mano que se irá ampliando durante el proceso de aprendizaje.

Uno de estos algoritmos más conocidos es el **algoritmo de Yarowsky,** que permite aprender un clasificador para una palabra ambigua (Yarowsky, 1995). El algoritmo de Yarowsky parte de un conjunto pequeño de datos de entrenamiento etiquetados (Λ_0) que contiene instancias para cada sentido de la palabra y un corpus no etiquetado mucho más grande (V_0).

En la primera iteración, el algoritmo de Yarowsky entrena un clasificador utilizando las instancias etiquetadas de Λ_0 y, a continuación, utiliza este clasificador para etiquetar el corpus no etiquetado V_0. El algoritmo selecciona entonces los ejemplos de V_0 para los que tiene mayor confianza en la clasificación, los elimina de V_0 (pasando a llamarse ahora el conjunto de datos no etiquetados V_1) y los añade al conjunto de datos de entrenamiento (que ahora pasa a llamarse Λ_1).

En la siguiente iteración, el algoritmo entrena un nuevo clasificador con Λ_1, utiliza el clasificador para etiquetar V_1 y extrae un nuevo conjunto de datos de entrenamiento Λ_2, y así sucesivamente. Con cada iteración, el corpus de los datos de entrenamiento Λ crece y el corpus de datos sin etiquetar V disminuye. El proceso se repite hasta que se alcanza una tasa de error suficientemente baja o hasta que no quedan más ejemplos no etiquetados.

El conjunto inicial de datos de entrenamiento Λ_0 se puede etiquetar a mano, escogiendo un conjunto de instancias aleatorias del corpus no etiquetado V_0 (Hearst, 1991) o se puede hacer uso de la ayuda de una heurística para seleccionar las mejores instancias (Yarowsky, 1995).

El algoritmo original de Yarowsky propone dos posibles heurísticas: la de un sentido por colocación y la de un sentido por discurso.

Heurística de un sentido por colocación

Se basa en la idea de que ciertas palabras o frases muy relacionadas con el sentido analizado no tienden a ocurrir con los otros sentidos. Entonces se define el conjunto de datos de entrenamiento escogiendo una única colocación para cada uno de los sentidos.

Heurística de un sentido por discurso

Se basa en la idea de que una palabra ambigua que aparece varias veces en un texto o discurso aparece a menudo con el mismo sentido (Gale, Church y Yarowsky, 1992). Esta heurística proporciona mejores resultados para los casos de homonimia que para los de polisemia (Krovetz, 1998) y es muy importante en el procesamiento del lenguaje porque muchas veces las tareas de desambiguación mejoran si se da un sesgo y se resuelve la ambigüedad de la misma manera en un mismo discurso.

Desambiguación basada en aprendizaje no supervisado

Los algoritmos de desambiguación basados en aprendizaje no supervisado aprenden los sentidos de las palabras a partir de un conjunto de datos de entrenamiento sin necesidad de disponer de definiciones para los sentidos que sean entendibles por parte de las personas. Estos algoritmos de desambiguación utilizan algún método estándar de agrupamiento, también llamado de *clustering*, y una medida de distancia para determinar la similitud entre clústeres. De hecho, el método más usado en tareas lingüísticas es el **agrupamiento aglomerativo** que va fusionando sucesivamente los clústeres más similares.

El **algoritmo de Schütze** (1992) (1998) representa cada palabra como un vector de contexto para una bolsa de palabras y, entonces, entrena el algoritmo siguiendo tres pasos:

5. Para cada aparición w_i de la palabra *w* en un corpus, se calcula un vector contexto *c*.
6. Se utiliza un algoritmo de agrupamiento para agrupar los vectores de contexto *c* en un número predefinido de clústeres. De hecho, cada uno de los clústeres define un sentido de la palabra *w*.
7. Se calcula el centroide de cada clúster. Cada centroide s_j es un vector que representa un sentido de la palabra *w*.

Dado que este es un algoritmo de clasificación no supervisado, no se conoce el nombre de cada uno de los sentidos de la palabra *w*; por lo que simplemente se les llama «el sentido *j* de la palabra *w*».

Para eliminar la ambigüedad de una instancia particular *t* de la palabra *w* utilizando el algoritmo de Schütze se aplican otros tres pasos:

1. Se calcula el vector contexto *c* para la instancia t.

2. Se recuperan todos los vectores de sentido s_j de la palabra *w*.
3. Se asigna *t* al sentido representado por el vector de sentido s_j que está más cerca *t*.

Accede a los ejercicios de autoevaluación a través del aula virtual

8.5. Similitud entre palabras

Accede al vídeo «Similitud entre palabras» a través del aula virtual

Una de las relaciones entre palabras que más se usa en el procesamiento del lenguaje natural es la sinonimia. La sinonimia es una relación binaria entre dos palabras: las palabras son sinónimas o no lo son. Sin embargo, se puede utilizar una métrica más relajada que permita calcular la similitud entre palabras, llamada **distancia semántica**.

Dos palabras son más **similares** si comparten más características de su significado, es decir, si son casi sinónimos. Por el contrario, dos palabras son menos similares o tienen mayor distancia semántica si comparten menos elementos de su significado.

Aunque se hable de similitud entre palabras, realmente la similitud debe medirse entre los sentidos de las palabras. Además, se debe distinguir entre **similitud entre palabras** y **relación entre palabras**. Dos palabras son similares si son casi sinónimos y una puede sustituir a la otra en un contexto dado, sin embargo, algunas palabras pueden estar relacionadas sin ser similares.

Por ejemplo, las palabras *coche* y *gasolina* están estrechamente relacionadas, pero no son similares; mientras que las palabras *coche* y *bicicleta,* además de estar

relacionadas, son más similares. De hecho, los antónimos son palabras que están relacionadas, pero que no son similares en absoluto. Es importante destacar que muchos de los algoritmos que calculan la similitud entre palabras, lo que en realidad hacen es utilizar una medida de relación entre palabras y no únicamente de similitud, aunque típicamente siguen llamándose **medidas de similitud**.

Para medir la similitud entre palabras o, mejor dicho, la relación entre sentidos de las palabras, existen dos tipos de algoritmos.

- Los primeros calculan la similitud entre palabras utilizando la estructura de un tesauro y se estudian en este tema.
- Los segundos calculan la similitud entre palabras aplicando métodos de distribución y encontrando directamente palabras que tienen distribuciones similares en un corpus.

Similitud entre palabras basada en tesauros

Los algoritmos que calculan la similitud entre palabras utilizando tesauros miden la distancia entre dos sentidos en un tesauro en línea como, por ejemplo, en WordNet. Estos algoritmos utilizan la estructura jerárquica del tesauro para definir la similitud entre palabras.

En principio, se podría medir la similitud utilizando cualquier **conocimiento** existente en el tesauro, como podría ser la meronimia o el glosario. Sin embargo, en la práctica, los algoritmos de similitud entre palabras basados en tesauros utilizan en general solo la jerarquía de relaciones de hiperonimia/hiponimia.

Los algoritmos más simples que utilizan tesauros se basan en la hipótesis de que los sentidos de las palabras son más similares si existe un camino más corto entre ellos en la representación gráfica del tesauro (Quillian, 1969). Entonces, un sentido es lo

más parecido a sí mismo, luego a sus padres o hermanos y luego, es menos similar a los sentidos de las palabras que están lejos en el gráfico.

Por lo que se define la **longitud del camino entre dos sentidos,** representados por los nodos c_1 y c_2, del gráfico del tesauro como $pathlen(c_1, c_2)$ y se calcula añadiendo una unidad al número de aristas existentes en el camino más corto entre los nodos c_1 y c_2.

Entonces, se define la medida de **similitud basada en la longitud del camino** como:

$$sim_{path(c_1,c_2)} = \frac{1}{pathlen(c_1, c_2)}$$

Sin embargo, en la mayoría de las aplicaciones no se dispone de datos de entrada con los sentidos etiquetados, por lo que es necesario que el algoritmo de similitud pueda proporcionar la similitud entre las palabras en lugar de entre sentidos. Se puede aproximar dicha similitud entre palabras (requeriría de un proceso de desambiguación de sentidos) utilizando el par de sentidos de las propias palabras, que son las que dan el máximo valor de similitud de los sentidos (Resnik, 1995). Por lo que formalmente se define la **similitud entre palabras** a partir de la similitud entre sentidos de la siguiente forma:

$$wordsim(w_1, w_2) = \max_{\substack{c_1 \in senses(w_1) \\ c_2 \in senses(w_2)}} sim(c_1, c_2)$$

El algoritmo que mide la similitud basada en la longitud del camino hace la suposición implícita de que cada eslabón, en la representación gráfica del tesauro, presenta una distancia uniforme. En la práctica, este supuesto no es apropiado porque las relaciones en un nivel más profundo de la jerarquía representan a menudo una distancia más cercana, mientras que otras relaciones más arriba en la jerarquía representan una distancia más amplia. La solución pasa por normalizar las distancias

en función a la profundidad de la jerarquía (Wu y Palmer, 1994) o asociar una distancia diferente a cada una de las aristas del tesauro.

Otros algoritmos que utilizan tesauros para calcular la similitud entre palabras se basan en la estructura del tesauro, pero también incluyen información probabilística derivada de un corpus. Son los llamados algoritmos de **similitud entre palabras basados en la cantidad de información** y utilizan medidas de teoría de la información para extraer información del corpus.

Según Resnik (1995), se define *P(c)* como la probabilidad de que una palabra seleccionada al azar en un corpus sea una instancia del concepto *c*, es decir, una variable aleatoria de las palabras asociadas con cada concepto. Cualquier palabra del tesauro es un descendiente del concepto de raíz, llamado *root*, lo que implica que $P(root) = 1$. Intuitivamente, cuanto más bajo se encuentre un concepto en la jerarquía, menor será su probabilidad.

Entonces, para calcular las probabilidades, se hace un recuento del corpus y cada palabra en el corpus cuenta como una ocurrencia de cada concepto que sea su ancestro.

Esto formalmente se define como:

$$P(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$$

Donde:

- $words(c)$ es el conjunto de palabras descendientes del concepto *c*.
- *N* es el número total de palabras en el corpus que también están presentes en el tesauro.

A partir de la teoría de la información, se define la **cantidad de información** (IC) de un concepto *c* como:

$$IC(c) = -\log P(c)$$

Además, a partir de la teoría de grafos se utiliza la definición de **ancestro común más bajo** de dos conceptos (LCS, *lowest common subsumer* en inglés). El $LCS(c_1, c_2)$ es el nodo más bajo en la jerarquía que tiene como descendientes a los conceptos c_1 y c_2, es decir que es el ancestro (padre, abuelo…) más cercano de ambos conceptos.

Existen diferentes medidas de similitud entre palabras calculadas a partir de la cantidad de información de un nodo. De hecho, la similitud entre dos palabras está relacionada con la información mutua entre las palabras, cuanta más información compartan las dos palabras, más similares van a ser. Resnik (1995) propone estimar la información mutua entre dos palabras como la cantidad de información de del ancestro común más bajo de los dos nodos que representan las palabras. Por lo tanto, la **medida de similitud de Resnik** se calcula con la siguiente fórmula:

$$sim_{resnik(c_1,c_2)} = -\log P\big(LCS(c_1, c_2)\big)$$

Otras medidas de similitud que amplían las ideas de Resnik son la medida de similitud de Lin (Lin, 1998) y la distancia de Jiang-Conrath (Jiang y Conrath, 1997).

Accede a los ejercicios de autoevaluación a través del aula virtual

8.6. Referencias bibliográficas

Agirre, E., López de Lacalle, O. y Soroa, A. (2014). Random walks for knowledge-based word sense disambiguation. *Computational Linguistics, 40*(1), 57-84.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Estados Unidos: MIT Press.

Gale, W. A., Church, K.W. y Yarowsky, D. (1992). One sense per discourse. En *Proceedings DARPA Speech and Natural Language Workshop* (pp. 233-237). New Jersey, Estados Unidos.

Hearst, M. A. (1991). Noun homograph disambiguation. *Proceedings of the 7th Conference of the University of Waterloo Centre for the New OED and Text Research*, 1-19.

Jiang, J. J. y Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. En *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*. Taiwan: ACL.

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics*. New Jersey, Estados Unidos: Prentice-Hall.

Kilgarriff, A. y Rosenzweig, J. (2000). Framework and results for English SENSEVAL. *Computers and the Humanities*, 34, 15-48.

Krovetz, R. (1998). *More than one sense per discourse*. Princeton, Estados Unidos: NEC Labs America.

Lesk, M. E. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. En *Proceedings of the 5th International Conference on Systems Documentation* (pp. 24-26). Nueva York, Estados Unidos: Association for Computing Machinery.

Lin, D. (1998). An information-theoretic definition of similarity. En *Proceeding ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 296-304). San Francisco, Estados Unidos: Morgan Kaufmann Publishers.

Mihalcea, R. (2007). Using wikipedia for automatic word sense disambiguation. *NAACL-HLT 07*, 196-203.

Navigli, R. y Lapata, M. (2010). An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(4), 678–692.

Ponzetto, S. P. y Navigli, R. (2010). Knowledge-rich word sense disambiguation rivaling supervised systems. En *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 1522-1531). Uppsala, Suecia: Association for Computational Linguistics.

Quillian, M. R. (1969). The teachable language comprehender: A simulation program and theory of language. *Communications of the ACM, 12*(8), 459-476.

RAE. (S. f.d). Homófono. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=KbZUpzR

RAE. (S. f.b). Homógrafo. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=Kbl2I5o

RAE. (S. f.c). Homónimo. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=KbriIov

RAE. (S. f.e). Metonimia. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=P7kP7xl

RAE. (S. f.a). Semántica. En *Diccionario de la lengua española* (actualización de la 23ª ed.). Recuperado de http://dle.rae.es/?id=XVRDns5

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxanomy. *International Joint Conference for Artificial Intelligence (IJCAI-95)* (pp. 448-453).

Schütze, H. (1992). Dimensions of meaning. En *Proceedings of Supercomputing '92* (pp. 787-796). Minneapolis, Estados Unidos: IEEE Press.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics, 24*(1), 97-124.

Vasilescu, F., Langlais, P. y Lapalme, G. (2004). Evaluating variants of the lesk approach for disambiguating words. En *4th International Conference on Language Resources and Evaluation* (pp. 633-636). Lisboa, Portugal: ELRA.

Weaver, W. (1955). Translation. En W. N. Locke y A. D. Boothe (Eds.), *Machine Translation of Languages* (pp. 15-23). Cambridge, Estados Unidos: MIT Press.

Wu, Z. y Palmer, M. (1994). Verb semantics and lexical selection. En *Proceedings of the 32th Annual Meetings of the Associations for Computational Linguistics (ACL-94*) (pp. 133-138). Las Cruces, México: ACL.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *ACL-95*, 189-196.

Yuret, D. (2004). Some experiments with a Naive Bayes WSD system. *Senseval-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Recuperado de http://www.aclweb.org/anthology/W04-0864

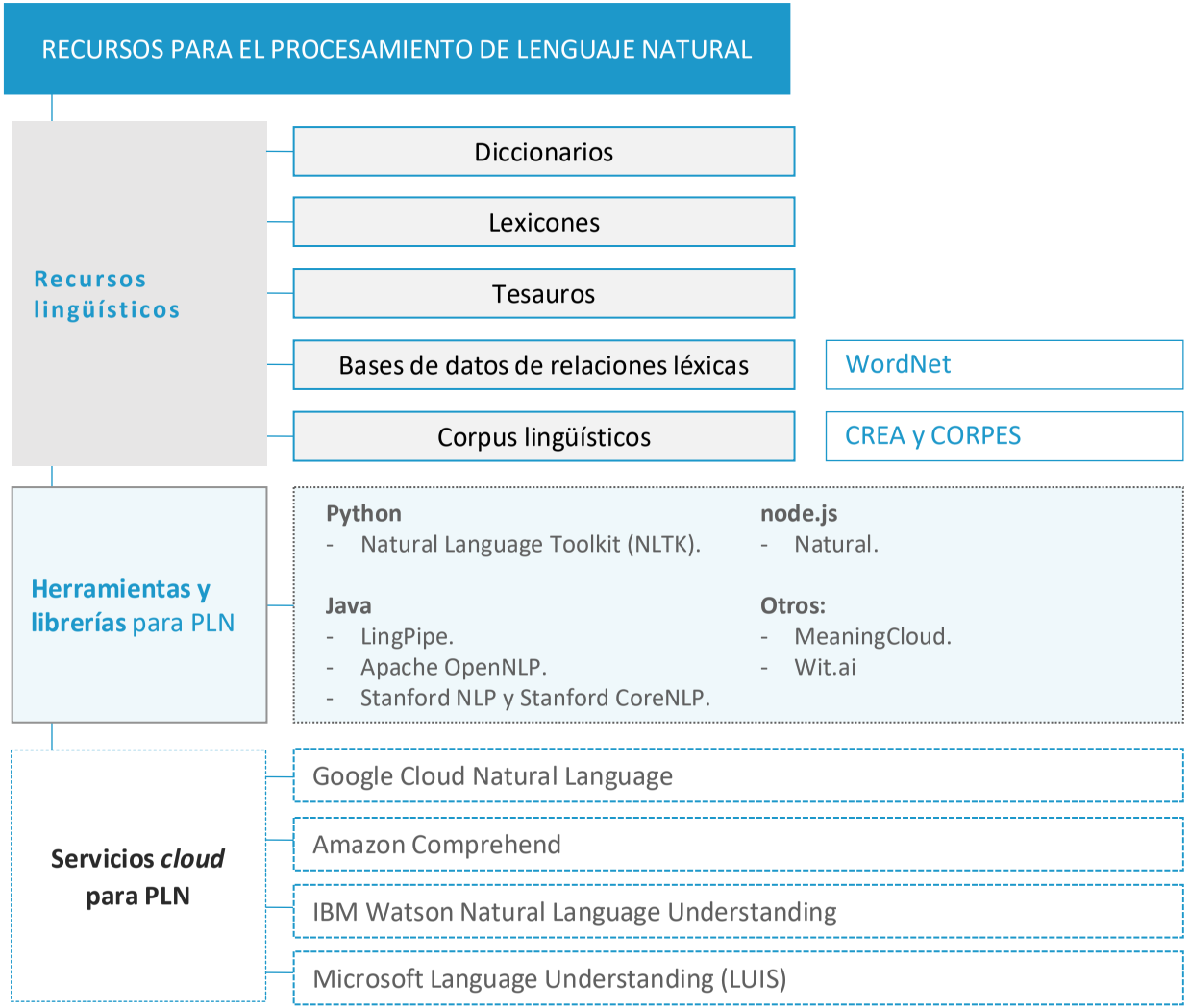
Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. En *Proceedings of the ACL 2010 System Demonstrations* (pp. 78-83). Uppsala, Suecia: Association for Computational Linguistics.

Accede al vídeo «Resumen» a través del aula virtual

Tema 9. Recursos para el procesamiento del lenguaje natural

Índice

Esquema



Ideas clave

9.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Los apartados de los que consta este tema son:

- Recursos lingüísticos.
- WordNet.
- Corpus en español.
- Herramientas y librerías para el procesamiento del lenguaje natural.
- Servicios *cloud* para el procesamiento del lenguaje natural.
- Referencias bibliográficas.

Al finalizar el estudio de este tema serás capaz de:

- Describir los diferentes tipos de recursos lingüísticos necesarios para implementar tareas de procesamiento del lenguaje natural.
- Analizar WordNet, una base de datos de relaciones léxicas.
- Analizar algunos corpus en español, por ejemplo, CREA y CORPES XXI.
- Conocer y utilizar las herramientas y librerías disponibles para implementar tareas de procesamiento del lenguaje natural en Python.
- Conocer los servicios *cloud* disponibles para implementar tareas de procesamiento del lenguaje natural.

Accede a los ejercicios de autoevaluación a través del aula virtual

9.2. Recursos lingüísticos

Accede al vídeo «Recursos lingüísticos» a través del aula virtual

La implementación de diferentes tareas de procesamiento del lenguaje natural, utilizando las técnicas y los algoritmos que se han estudiado en capítulos anteriores, requiere de una serie de recursos lingüísticos. Entre las bases de conocimiento lingüístico destacan:

- Los diccionarios.
- Los lexicones.
- Los tesauros.
- Las bases de datos de relaciones léxicas.

Además, de estas bases de conocimiento lingüístico también cabe destacar las colecciones de textos o de recursos del habla llamados corpus.

A continuación, se recogen las definiciones de los diferentes tipos de recursos lingüísticos y, en las siguientes secciones, se presentan la base de datos de relaciones léxicas WordNet y algunos corpus en español.

> **Diccionario:** es el repertorio donde se recogen, según un orden determinado, las palabras de una lengua acompañadas de su definición o explicación. Los diccionarios incluyen una descripción detallada y comprensible para un humano de los diferentes sentidos de las palabras.

> **Lexicón:** un diccionario que contiene información morfológica. Este diccionario morfológico es un repertorio donde se recogen una lista de morfemas y la información básica sobre estos.

Tesauro, también llamado *thesaurus*, *thesauri* o tesoro: se refiere al diccionario que contiene una lista de significados de las palabras y las relaciones entre estos significados.

- En la literatura, un tesauro contiene una lista de palabras con sus sinónimos y sus antónimos.
- En lingüística, un tesauro recoge el conocimiento sobre las relaciones de hiperonimia/hiponimia y meronimia/holonimia representadas en la propia estructura jerárquica del tesauro. Es decir que la jerarquía del tesauro modela la relación is-a y la relación parte-todo de los sentidos de las palabras. Además, un tesauro también puede recoger conocimiento sobre otras relaciones semánticas como la sinonimia o antonimia.

Bases de datos de relaciones léxicas: una generalización de los tesauros, es decir, un tipo de diccionario que recoge conocimiento sobre cualquier relación semántica entre sentidos de las palabras. Estas bases contienen un conjunto de lemas, cada uno de los cuales está anotado con el posible conjunto de sentidos de la palabra y las relaciones entre esos sentidos.

Corpus lingüístico: es una colección de textos representativos de una lengua que se utilizan para el análisis lingüístico. Se puede distinguir entre:

- Los corpus textuales, que recogen extractos de libros, revistas, periódicos o cualquier otra fuente escrita.
- Los corpus orales que son colecciones de habla, es decir extractos de audios provenientes de fuentes como puede ser la radio o la televisión.

Los corpus pueden estar anotados o etiquetados de forma que las palabras que lo conforman presentan, además, algún tipo de información lingüística, por ejemplo, información sintáctica, semántica o pragmática, son los conocidos como **corpus etiquetados**.

Los primeros corpus disponibles *online* aparecieron en la década de 1960. Uno de los pioneros fue el **Brown Corpus**, un corpus del inglés americano desarrollado en 1963

por la Brown University y que contenía una colección de un millón de palabras extraídas de 500 textos de diferentes géneros: periódicos, novelas, no ficción, académico, etc. (Kucera y Francis, 1967). Al principio el corpus no estaba etiquetado, pero con el paso de los años se etiquetó con información sobre las categorías gramaticales (POS).

Hoy en día los corpus tienden a ser mucho más extensos que el Brown Corpus. Por ejemplo, el corpus etiquetado con información morfosintáctica **WSJ**; se trata de la colección del millón de palabras que se publicaron en los artículos del Wall Street Journal (WSJ) en el año 1989.

Accede a la primera versión del WSJ desde la siguiente dirección web:

https://catalog.ldc.upenn.edu/LDC2000T43

Accede a los ejercicios de autoevaluación a través del aula virtual

9.3. WordNet

Accede al vídeo «WordNet» a través del aula virtual

WordNet es una gran base de datos léxica en inglés (Fellbaum, 1998) donde los sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos llamados *synsets*, y cada uno expresa un sentido distinto. Contiene 117 000 *synsets* que están interrelacionados por medio de relaciones conceptuales semánticas y léxicas.

Accede a WordNet a través del aula virtual o desde la siguiente dirección web:

https://wordnet.princeton.edu/

WordNet se asemeja a un tesauro en el sentido de que agrupa palabras en función de sus significados, sin embargo, interconecta los sentidos específicos de las palabras de forma que se puedan desambiguar semánticamente. Además, etiqueta los tipos de relaciones semánticas entre palabras, mientras que en un tesauro solo se contempla la similitud entre estas y no se explicita tipo de relación. Por lo tanto, WordNet recoge explícitamente relaciones de sinonimia y antonimia, hiperonimia/hiponimia y meronimia.

La red de relaciones entre palabras y sentidos de WordNet es accesible *online* y se puede navegar a través de ella, tal como se observa en la figura 1. Además, WordNet se puede descargar de forma gratuita bajo una licencia de *software* libre. La última versión la 3.1 y está disponible para su consulta *online* desde noviembre de 2012.

Consulta WordNet versión 3.1 desde la siguiente dirección web:
http://wordnetweb.princeton.edu/perl/webwn

Sin embargo, la versión más reciente para descargar para sistemas Windows es la versión 2.1 y para sistemas UNIX, Linux, Mac OS X y Solaris es la versión 3.0

Consulta las diferentes versiones de descarga en la siguiente dirección web:
https://wordnet.princeton.edu/download

En la figura 1 podemos ver el resultado de la búsqueda en WordNet de la palabra «banco» donde se observa la relación de hiponimia (*direct hyponym*) para el *synset* (S) conformado por los nombres *depositary financial institution*, *bank*, *banking concern* y *banking Company.*

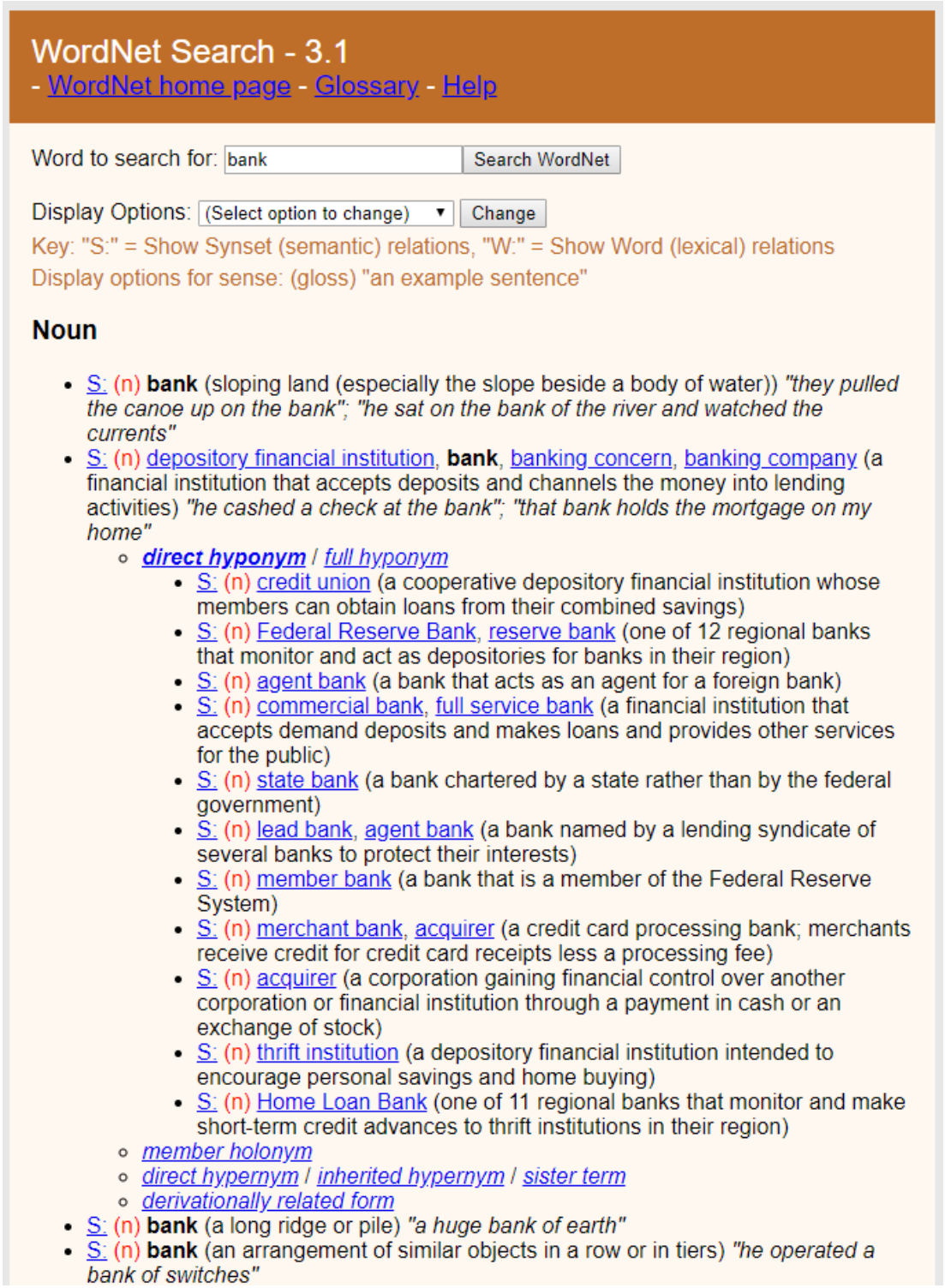


Figura 1. Resultados de la búsqueda en WordNet de la palabra «banco».

Fuente: http://wordnetweb.princeton.edu/perl/webwn

La estructura de WordNet la convierte en una herramienta muy útil por lo que es una de las más utilizadas en la lingüística computacional y en el procesamiento del lenguaje natural. Es por eso por lo que se han creado diferentes recursos similares a

WordNet para otras lenguas, por ejemplo, existe una versión de WordNet en español (Fernández, Vázquez y Fellbaum, 2008).

Para completar el estudio de esta sección deberás leer la sección **2.4.2 WordNet** de:

Moreno, A. (2000). WordNet. En *Autor, Diseño de implementación de un lexicón computacional para lexicografía y traducción automática* (Vol. 9). Recuperado de http://elies.rediris.es/elies9/2-4-2.htm

Accede a los ejercicios de autoevaluación a través del aula virtual

9.4. Corpus en español

Accede al vídeo «Herramientas y librerías para el procesamiento del lenguaje natural» a través del aula virtual

En las secciones anteriores se ha comentado sobre algunos corpus, esas colecciones de textos para el análisis lingüístico, utilizados en el procesamiento del lenguaje natural en inglés. En el ámbito hispánico son instituciones como la Real Academia Española, incluida en la Asociación de Academias de la Lengua Española, o el Instituto Cervantes las que se han ocupado de producir recursos lingüísticos que puedan ser utilizados en el procesamiento del lenguaje natural en español.

La Biblioteca Virtual Miguel de Cervantes recoge algunos corpus en español, por ejemplo:

- Corpus de sonetos del siglo de oro: http://goldenage.cervantesvirtual.com/
- Corpus diacrónico del español histórico IMPACT-es (documentación)**:** http://data.cervantesvirtual.com/blog/documentacion-corpus-diacronico/

El **Corpus diacrónico del español histórico IMPACT-es** (Sánchez-Martínez, Martínez-Sempere, Ivars-Ribes y Carrasco, 2013) contiene 86 obras pertenecientes a la Biblioteca Virtual Miguel de Cervantes e impresas entre los años 1482 y 1647. Este corpus viene acompañado por un lexicón que enlaza más de 10 000 lemas con las diferentes variantes de las palabras encontradas en los documentos. Además, las palabras más frecuentes del corpus están anotadas con su lema, categoría gramatical y su forma moderna equivalente. Esto permite efectuar búsquedas de forma muy eficiente (Carrasco et al., 2015) a través del interfaz web.

- Accede al IMPACT-es: http://data.cervantesvirtual.com/blog/diasearch/
- Disponible para descarga: https://www.digitisation.eu/tools-resources/language-resources/impact-es/

Un ejemplo de búsqueda se presenta en la figura 2 donde vemos el resultado de la búsqueda en el corpus diacrónico del español histórico (IMPACT-es) de obras literarias en las que aparece una expresión del tipo «hacer algo de», donde el verbo *hacer* puede aparecer en cualquiera de sus formas, como en español antiguo, y la palabra «algo» puede ser cualquier nombre, como por ejemplo «haz examen de» o «haciendo burla de».

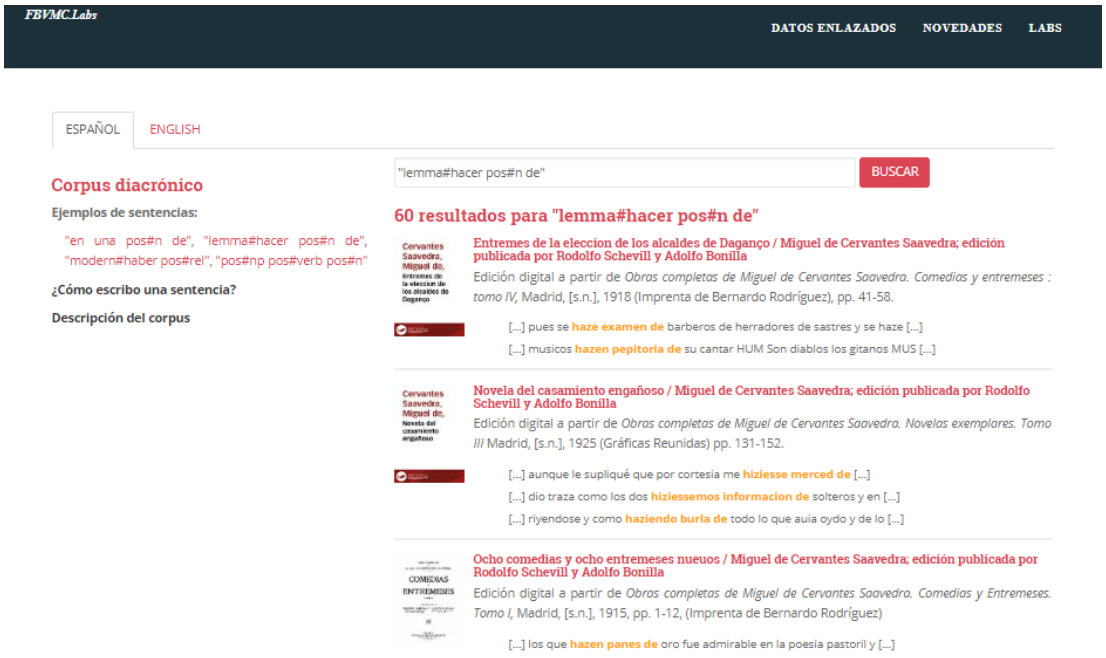


Figura 2. Resultado de la búsqueda en el corpus diacrónico del español histórico (IMPAC-es) de obras literarias en las que aparece una expresión del tipo «hacer algo de».
Fuente: http://data.cervantesvirtual.com/blog/diasearch/

La Real Academia Española se fijó a finales del siglo pasado el objetivo de generar recursos lingüísticos para mostrar la evolución del español en distintas épocas y su funcionamiento actual en los diferentes países que conforman la comunidad lingüística hispánica. Es por eso por lo que, a partir de 1996, la Real Academia Española en colaboración con las otras academias de la Asociación de Academias de la Lengua Española crearon el **Corpus de Referencia del Español Actual (CREA)** y el **Corpus del Español del Siglo XXI (CORPES XXI)**.

Para completar el estudio de esta sección deberás leer el capítulo 7.1 **CREA y CORPES, corpus de referencia del español** del siguiente libro:

Sánchez, M. (2016). CREA y CORPES, corpus de referencia del español. En A. L. Gonzalo (Coord.), *Tecnologías del lenguaje en España: comunicación inteligente entre personas y máquinas* (pp. 119-124). Barcelona: Fundación Telefónica y Ariel. Recuperado de https://www.fundaciontelefonica.com/arte_cultura/publicaciones-listado/pagina-item-publicaciones/itempubli/565/

Accede a los ejercicios de autoevaluación a través del aula virtual

9.5. Herramientas y librerías para el procesamiento del lenguaje natural

Accede al vídeo «Herramientas y librerías para el procesamiento del lenguaje natural» a través del aula virtual

Existen numerosas herramientas y librerías que facilitan el desarrollo e implementación de diferentes tareas relacionadas con el procesamiento del lenguaje natural.

Una de las herramientas más utilizadas para la implementación en Python de aplicaciones para el procesamiento del lenguaje natural es **Natural Language Toolkit (NLTK)**.

NLTK está disponible para Windows, Mac OS X y Linux, es *open source* y su código se distribuye bajo la licencia Apache License Version 2.0.

Accede a NLTK a través del aula virtual o desde la siguiente dirección web:
http://www.nltk.org/

NLTK es una plataforma que proporciona interfaces a más de 50 corpus y recursos léxicos, como por ejemplo WordNet, además de una serie de librerías para realizar tareas de clasificación, obtención de *tokens*, derivación, etiquetado morfosintáctico, análisis sintáctico y análisis semántico.

A continuación veremos varios ejemplos de la utilización de NLTK.

En la figura 3 vemos el código Python para el **etiquetado morfosintáctico** de la oración «*They refuse to permit us to obtain the refuse permit*» utilizando NLTK.

```
>>> text = word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

Figura 3. Etiquetado morfosintáctico de una oración con ambigüedades.
Fuente: Bird, Klein, y Loper, 2014

En la siguiente figura vemos código Python para el **análisis sintáctico** de la oración «*I shot an elephant in my pajamas*» utilizando NLTK.

```
>>> groucho_grammar = nltk.CFG.fromstring("""
... S -> NP VP
... PP -> P NP
... NP -> Det N | Det N PP | 'I'
... VP -> V NP | VP PP
... Det -> 'an' | 'my'
... N -> 'elephant' | 'pajamas'
... V -> 'shot'
... P -> 'in'
... """)
```

```
>>> sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
>>> parser = nltk.ChartParser(groucho_grammar)
>>> for tree in parser.parse(sent):
...     print(tree)
...
(S
  (NP I)
  (VP
    (VP (V shot) (NP (Det an) (N elephant)))
    (PP (P in) (NP (Det my) (N pajamas)))))
(S
  (NP I)
  (VP
    (V shot)
    (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
```

Figura 4. Análisis sintáctico de una oración ambigua.
Fuente: Bird, Klein, y Loper, 2014

NLTK ha sido desarrollado principalmente para el procesamiento del lenguaje natural en inglés. Sin embargo, esta plataforma tiene la flexibilidad necesaria para poder realizar procesamiento del lenguaje natural en otros idiomas como el español. Por ejemplo, NLTK se puede utilizar para entrenar un etiquetador morfosintáctico en español a partir de un corpus etiquetado y también permite incorporar un etiquetador externo que sea capaz de analizar el español.

Además de este recurso en Python, cabe destacar algunas **librerías para el desarrollo en Java** de aplicaciones en el ámbito del procesamiento del lenguaje natural:

- LingPipe: http://alias-i.com/lingpipe/index.html
- Apache OpenNLP: https://opennlp.apache.org/docs/
- Stanford NLP: http://nlp.stanford.edu/software/
- Stanford CoreNLP: https://stanfordnlp.github.io/CoreNLP/

Stanford CoreNLP incorpora modelos para realizar tareas de procesamiento del lenguaje natural en español. Por lo tanto, permite gestionar algunas características típicas de este idioma, como por ejemplo la separación de contracciones de palabras (*del = de + el*) durante el proceso de obtención de *tokens* o la identificación de los tiempos y modos de los verbos (*presente de indicativo*) en el etiquetado morfosintáctico.

Stanford CoreNLP está implementado en Java, pero proporciona acceso por línea de comandos lo que permite su integración en Python. Esta característica es muy útil si se quiere utilizar el etiquetado morfosintáctico para el español del Stanford CoreNLP en un código desarrollado en Python utilizando la plataforma NLTK.

En el caso de querer realizar una implementación de tareas de procesamiento del lenguaje natural en node.js se puede utilizar la librería Natural.

Accede a la guía de la librería Natural:
https://github.com/NaturalNode/natural/blob/master/README.md

Por último, cabe destacar las siguientes **herramientas**:

- MeaningCloud: es una API a través de la cual se puede realizar el análisis sintáctico o el etiquetado morfosintáctico en varios idiomas incluyendo el español. https://www.meaningcloud.com/developer/lemmatization-pos-parsing

- Wit.ai: es una herramienta que permite a los desarrolladores implementar *bots*, asistentes personales y aplicaciones móviles con las que se pueda interactuar a través del habla. Wit.ai proporciona la API de forma gratuita a cambio de que las aplicaciones desarrolladas con la misma sean abiertas. https://wit.ai/

Accede a los ejercicios de autoevaluación a través del aula virtual

9.6. Servicios *cloud* para el procesamiento del lenguaje natural

Accede al vídeo «Herramientas y librerías para el procesamiento del lenguaje natural» a través del aula virtual

Existen numerosos servicios *cloud* para el procesamiento del lenguaje natural. El auge de la inteligencia artificial y la proliferación de los *bots* conversacionales ha hecho que todas las grandes empresas quieran tener una API que permita el desarrollo de aplicaciones basadas en el procesamiento del lenguaje natural. Entre estos servicios en la nube podemos destacar los siguientes:

- Google Cloud Natural Language: https://cloud.google.com/natural-language/
- Amazon Comprehend: https://aws.amazon.com/comprehend/
- IBM Watson Natural Language Understanding: https://www.ibm.com/watson/services/natural-language-understanding/
- Microsoft Language Understanding (LUIS): https://www.luis.ai/

Google Cloud Natural Language

Google Cloud Natural Language es un servicio que permite revelar la estructura y el significado del texto en diferentes idiomas incluyendo el inglés y el español. Se ofrece

a través de una API REST de pago, pero se permite su uso gratuito para analizar pequeños volúmenes de datos.

También permite realizar el análisis sintáctico de una oración identificando las relaciones gramaticales establecidas entre las palabras, por lo que muestra el resultado del análisis sintáctico en un formato de dependencias. Para realizar el análisis sintáctico, se identifican las diferentes partes de la oración y se obtienen el lema y las características morfológicas de las palabras. Por lo tanto, Google Cloud Natural Language facilita el análisis morfológico de una oración y etiquetarla morfosintácticamente.

Además de las tareas análisis, puede extraer información acerca de personas, lugares y eventos mencionados en documentos de texto, comprender el sentimiento en una opinión acerca de un producto en las redes sociales o analizar la intención de una conversación con clientes en una llamada.

Su funcionamiento se basa en modelos de aprendizaje automático y, además, permite crear modelos personalizados para el procesamiento del lenguaje natural a través de AutoML Natural Language. Esta herramienta permite subir un corpus de documentos de texto, etiquetarlos, utilizarlos para entrenar un modelo y evaluar el modelo creado.

Amazon Comprehend

Amazon Comprehend es un servicio de procesamiento de lenguaje natural que utiliza técnicas de aprendizaje automático para buscar ideas y relaciones en el texto. Es un servicio de pago con base en el uso, aunque se permite utilizar la capa gratuita durante 12 meses para analizar pequeños volúmenes de datos.

Identifica el idioma del texto; extrae frases clave, lugares, personas, marcas o eventos; entiende cuán positivo o negativo es el texto; analiza el texto usando extracción de *tokens* y categorías gramaticales y organiza automáticamente una

colección de archivos de texto en función de su tema. También permite analizar texto y aplicar los resultados en diferentes aplicaciones como, por ejemplo, el análisis de voz de los clientes, la búsqueda inteligente de documentos y la personalización del contenido para aplicaciones web.

Amazon Comprehend aprende para mantenerse al día con la evolución del lenguaje. El aprendizaje lo basa en una gran variedad de fuentes de información, como por ejemplo las descripciones de productos de Amazon.com y las reseñas que los consumidores hacen de estos productos.

IBM Watson Natural Language Understanding

IBM Watson Natural Language Understanding es un servicio de procesamiento del lenguaje natural que permite realizar funciones avanzadas de análisis de texto. Es un servicio de pago del catálogo de IBM Watson, sin embargo, a través de IBM Cloud se permite su uso gratuito sin restricciones temporales.

Interpreta texto en trece lenguas diferentes, entre las que se incluye el español. Natural Language Understanding permite comprender datos no estructurados y extraer entidades, relaciones, palabras clave y roles semánticos. Por ejemplo, puede identificar el concepto de inteligencia artificial en un documento de investigación sobre aprendizaje profundo donde no aparece el término «inteligencia artificial» o detectar sentimientos en un texto a partir del contenido o del contexto.

Además, da la posibilidad de introducir conocimiento del propio dominio de aplicación como conocimiento de la organización o la industria para mejorar las tareas de procesamiento del lenguaje natural y desarrollar modelos personalizados.

Microsoft Language Understanding (LUIS)

Microsoft Language Understanding (LUIS) es un servicio basado en aprendizaje automático para identificar información valiosa en conversaciones. Es un servicio de pago con base en el uso, aunque se permite probar de forma gratuita las funcionalidades de procesamiento de texto, no las del habla.

Es capaz de interpretar los objetivos del usuario, es decir sus intenciones, y extraer información valiosa de las oraciones como, por ejemplo, las diferentes entidades, y funciona para el procesamiento del lenguaje natural en doce idiomas.

Asimismo, aplica los modelos obtenidos a través de aprendizaje automático para identificar el significado de conversaciones en lenguaje natural. Utiliza técnicas de aprendizaje por refuerzo para mejorar la calidad de los modelos y técnicas de aprendizaje activo para actualizar constantemente los modelos, incluso creando modelos propios o personalizados según las necesidades de los usuarios.

Está diseñado para habilitar tareas de procesamiento del lenguaje natural en aplicaciones móviles, *bots* y dispositivos del Internet de las cosas. Además, se integra con Azure Bot Service, el servicio para la creación de agentes conversacionales de Microsoft, lo que facilita su utilización para la creación de *bots*.

Accede a los ejercicios de autoevaluación a través del aula virtual

9.7. Referencias bibliográficas

Bird, S., Klein, E. y Loper, E. (2014). *Natural Language Processing with Python*. Sebastopol (Estados Unidos): O'Reilly.

Carrasco, R. C., Martínez-Sempere, I., Mollá-Gandía, E., Sánchez-Martínez, F., Candela-Romero, G. y Escobar-Esteban M. P. (2015). Linguistically-Enhanced Search over an Open Diachronic Corpus. En A. Hanbury, G. Kazai, A. Rauber y N. Fuhr (Eds.), Advances in Information Retrieval. ECIR 2015. Lecture Notes in Computer Science (Vol. 9022). https://doi.org/10.1007/978-3-319-16354-3_89

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge (Estados Unidos): MIT Press.

Fernández, A., Vázquez, G. y Fellbaum, C. (2008). The Spanish version of WordNet 3.0. En A. Storrer, A. Geyken, A. Siebert y K. M. Würzner (Eds.), *Text resources and lexical knowledge. Selected papers from the 9th conference on natural language processing KONVENS 2008*. Berlin: De Gruyter Mouton.

Kucera, H. y Francis, W. N. (1967). *Computational analysis of present-day American English*. Providence (Estados Unidos): Brown University Press.

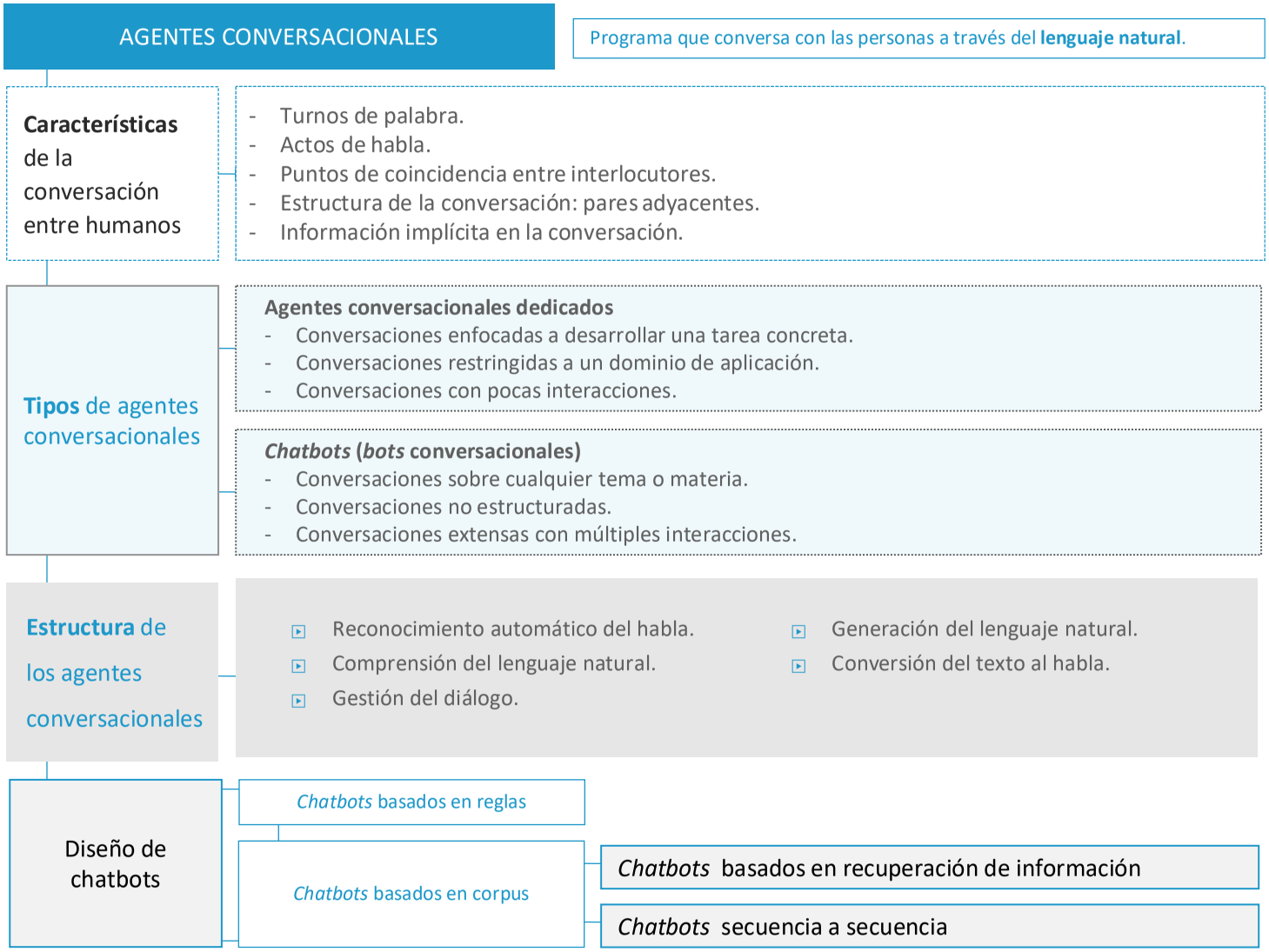
Sánchez-Martínez, F., Martínez-Sempere, I., Ivars-Ribes, X. y Carrasco, R.C. (2013). An open diachronic corpus of historical Spanish. *Language Resources and Evaluation 47*(4), 1327-1342.

Accede al vídeo «Resumen» a través del aula virtual

Tema 10. Agentes conversacionales

Índice

Esquema



Ideas clave

10.1. Introducción y objetivos

Accede al vídeo «Introducción y objetivos» a través del aula virtual

Este tema comienza definiendo lo que es un agente conversacional. Analizamos las características más relevantes de las conversaciones entre humanos que son las que determinan las características de los agentes conversacionales. Además, en este tema se describen los agentes conversacionales dedicados y los *chatbots*, dos tipos de agentes conversacionales que se diferencian por el tipo de conversación que son capaces de mantener y la funcionalidad que pretenden implementar. También presentamos la estructura básica de un agente conversacional y se describe la utilidad de cada uno de los módulos que la componen. Por último, veremos los diferentes enfoques de cómo diseñar un *chatbot*.

Los apartados de los que consta este tema son:

- Definición de agente conversacional.
- Características de las conversaciones entre humanos.
- Tipos de agentes conversacionales.
- Estructura de los agentes conversacionales.
- Diseño de *chatbots*.

Al finalizar el estudio de este tema serás capaz de:

- Definir lo que es un agente conversacional.
- Describir las características más relevantes de las conversaciones entre humanos que son las que determinan las características de los agentes conversacionales.

- Identificar los diferentes tipos de agentes conversacionales dependiendo del tipo de conversación que son capaces de mantener y la funcionalidad que pretenden implementar.
- Representar gráficamente la estructura básica de los agentes conversacionales y describir la utilidad de cada uno de los módulos que la componen.
- Describir los diferentes enfoques utilizados para diseñar un *chatbot*.

Accede a los ejercicios de autoevaluación a través del aula virtual

10.2. Definición de agente conversacional

Accede al vídeo «Características de las conversaciones entre humanos» a través del aula virtual

La Real Academia Española (RAE, s. f.) en su diccionario de la lengua española define la palabra **diálogo:** «Del lat. *dialŏgus*, y este del gr. *διάλογος diálogos*. 1. m. Plática entre dos o más personas, que alternativamente manifiestan sus ideas o afectos». Este mismo diccionario define la palabra **plática:** «De *práctica*.
1. f. conversación (‖ acción de hablar)».

Los **agentes conversacionales**, también llamados **sistemas de diálogo**, son programas que conversan con las personas a través del lenguaje natural.

Los agentes conversacionales pueden interactuar con el humano, ya sea a través de la voz (hablando con el usuario), de texto, en el caso de que la conversación se lleve a cabo a través de un chat de texto, o utilizando ambas modalidades a la vez. Los **agentes conversacionales multimodales** integran el habla con otras modalidades como los gráficos o las imágenes.

Algunos agentes conversacionales modernos están empezando a incorporar las emociones, por ejemplo: sorpresa, enfado, alegría, tristeza o estrés, en la conversión

de texto en habla para dar más realismo a la respuesta del agente conversacional. De hecho, las emociones también pueden formar parte del proceso de comprensión del lenguaje y la posterior generación del diálogo más adecuado a las emociones detectadas. Las emociones se pueden detectar del propio contenido del mensaje, pero también se podrían extraer del análisis de los gestos y expresiones faciales de la persona que interactúa con el agente conversacional.

En el caso más sencillo de agente conversacional, este es capaz de contestar a las preguntas del usuario o de realizar una acción de control a partir de un comando de voz. Este es el caso de los asistentes virtuales que tienen una única interacción con la persona. Sin embargo, también existen agentes conversacionales más complejos capaces de continuar con la interacción con el usuario y, por ejemplo, contestar a una segunda pregunta relacionada con la primera. Sería el caso de agentes conversacionales que ya incluyen diálogo y que se han aplicado en asistentes para planificar y gestionar viajes, como en el caso de GUS (Bobrow et al., 1977).

Los agentes conversacionales incluso pueden usarse en dominios mucho más complejos como los agentes conversacionales pedagógicos que actúan como tutor del estudiante, como sería el caso de ITSPOKE (Forbes-Riley y Litman, 2011). Por último, los agentes conversacionales se pueden usar simplemente como divertimento para charlar con ellos, pero sin que la idea sea realizar una tarea concreta, como en Cleverbot.

Accede a los ejercicios de autoevaluación a través del aula virtual

10.3. Características de las conversaciones entre humanos

Accede al vídeo «Tipos de agentes conversacionales» a través del aula virtual

Las principales características de los agentes conversacionales se deben a las características intrínsecas que tiene una conversación entre humanos. Una conversación es una acción que se realiza de forma compartida entre varias personas y que implica un alto grado de complexidad, ante esto, las conversaciones entre una persona y una máquina intentan emular los comportamientos que se dan en las conversaciones entre humanos.

Turnos de palabra

Una conversación se compone de **turnos de palabra** que se dan de forma consecutiva y que alternan las opiniones de los interlocutores. En el caso de que dos personas mantengan una conversación, el interlocutor A declara algo, el interlocutor B le responde, luego el interlocutor A vuelve a expresarse, y así sucesivamente.

Las personas somos capaces de identificar fácilmente quien es la siguiente persona a la que le toca expresarse y el momento exacto en el que debe intervenir. Por lo tanto, durante una conversación hablada, los humanos somos capaces de esperar a que nuestro interlocutor acabe de hablar antes de responder, identificar que nuestro interlocutor ya ha acabado su frase y saber que somos nosotros los que debemos responder, determinar el momento preciso en el que debemos responder (por ejemplo, un instante después de que acabe de hablar nuestro interlocutor), o interpretar que un silencio muy prolongado de nuestro interlocutor puede significar que le incomoda la pregunta que le hemos realizado y que no la va a contestar.

En las conversaciones entre humanos, los turnos de palabra están claros y se asignan de forma intuitiva, sin embargo, para los agentes conversacionales llegar a manejar correctamente los turnos de palabra y gestionar eficientemente los tiempos es una tarea complicada. La gestión de los turnos de palabra en los agentes conversacionales se estudia en el campo del **análisis conversacional** (Levinson, 1983) y se basa en la

idea de que el intercambio de turnos se puede modelar con una serie de reglas. Estas reglas, que indican cómo manejar los turnos de palabra, se evalúan en los momentos en que la estructura del lenguaje permite intercambiar el interlocutor, momentos llamados en inglés *transition-relevance places* (TRP) (Sacks, Schegloff y Jefferson, 1974).

Actos de habla

Se ha comentado en el apartado anterior que una conversación se compone de una secuencia de turnos de palabra. Además, se puede añadir que cada uno de los turnos incluye una o varias expresiones o declaraciones. La **teoría de los actos de habla** sugiere que cada expresión en una conversación es un tipo de acción que realiza el interlocutor (Austin, 1962). Esta teoría fue desarrollada por el filósofo británico John L. Austin antes de morir en 1960 y presentada en su obra póstuma, en la cual se acuñó el término **acto de habla**.

Un **acto de habla** es cada uno de los tipos de acciones que involucran el uso del lenguaje natural.

En una expresión tal como «a mi nuevo gato lo he llamado Felix», la acción de llamar a mi gato Felix tiene la consecuencia de que el gato tenga el nombre de Felix. En este caso queda claro que la acción que realiza el interlocutor y que se recoge en la expresión ha permitido cambiar el estado del mundo haciendo que el gato que no tenía nombre tenga ahora uno. Sin embargo, el trabajo de Austin sugiere que todas las expresiones tienen asociado algún acto de habla, aunque este no sea tan evidente como el descrito anteriormente.

Los actos de habla definidos por Austin se han ido extendiendo con el paso de los años y hoy en día podemos hablar de actos que cambian el estado del mundo, sugieren al interlocutor que haga algo, le dan directrices para que haga algo, prometen o planifican que el interlocutor va a hacer algo en el futuro o expresan el estado psicológico o la actitud del interlocutor sobre una acción.

En una conversación los humanos somos capaces de hacer preguntas, dar órdenes o informar sobre algo sin darnos cuenta de que nuestras expresiones contienen diferentes tipos de los actos de habla definidos por Austin. Sin embargo, para los agentes conversacionales no es trivial saber interpretar en cada momento en qué acto de habla se encuentran y tomar las decisiones correctas para responder de la forma más adecuada.

No se puede afirmar que todos los agentes conversacionales sean capaces de tratar con múltiples de actos de habla. De hecho, solo algunos agentes conversacionales modernos implementan funcionalidades más avanzadas y son capaces de lanzar una pregunta, dar una orden, hacer una propuesta, rechazar una sugerencia o estar de acuerdo con una expresión presentada por su interlocutor.

Puntos de coincidencia entre interlocutores

Se ha explicado en el apartado anterior que cada turno de palabra y, más en concreto, cada expresión del turno se puede ver como una acción de un interlocutor. Sin embargo, un diálogo no se compone de una serie de actos independientes sin relación alguna. Sino que, a diferencia de en un monólogo, una conversación es un acto colectivo realizado entre ambos interlocutores. Debido a esta acción conjunta que es una conversación, se deben establecer constantemente unos **puntos de coincidencia** entre los interlocutores (Stalnaker, 1978).

Los **puntos de coincidencia** son el conjunto de fundamentos en los que creen los interlocutores que participan en una conversación.

Es necesario asegurar que se han establecido unos puntos de coincidencia entre interlocutores para que la conversación pueda desarrollarse con éxito. Esto significa que el receptor del mensaje debe dejar claro que ha entendido el significado y la intención de la expresión lanzada por el emisor, y también el caso contrario, indicar que no ha entendido al emisor del mensaje (Clark, 1996).

En las conversaciones entre humanos se utilizan diferentes técnicas para garantizar que se están estableciendo los puntos de coincidencia entre interlocutores (Clark y Schaefer, 1989). En una conversación, el receptor del mensaje puede asentir o utilizar palabras tipo «vale» o «genial» u onomatopeyas tipo «ajá» para mostrar que está entendiendo y, además, está de acuerdo con su interlocutor.

Otras opciones que garantizan los puntos de coincidencia serían que el receptor del mensaje reproduzca textualmente, reformule o parafrasee lo que le ha transmitido su interlocutor o que le ayude a acabar de construir su argumentación. Además, el receptor del mensaje puede mostrar interés por la conversación y entonces se interpreta directamente que está conforme con lo que le comenta su interlocutor. Por último, el receptor puede simplemente pasar a proponer su siguiente contribución al diálogo, por lo que se supone que está de acuerdo con lo que se ha comentado hasta ese momento.

Es imprescindible que los agentes conversacionales imiten todos los comportamientos que se acaban de describir para poder asegurar que se establecen puntos de coincidencia entre el agente y su interlocutor humano. De hecho, se ha demostrado que si el agente no garantiza que ha entendido al humano, va a crear confusión a la persona y esta no se va a sentir cómoda utilizando el agente conversacional (Yankelovich, Levow y Marx, 1995).

> **Ejemplo ilustrativo 1.** Posibles conversaciones de un asistente virtual que ayuda a un cliente de una operadora de telefonía a configurar su perfil de usuario, consultar su consumo…
>
> Conversación A:
> *Asistente: ¿Desea revisar algo más de su perfil de usuario?*
> *Cliente: No.*
> *Asistente: ¿Qué desea hacer ahora?*
>
> Conversación B:
> *Asistente: ¿Desea revisar algo más de su perfil de usuario?*
> *Cliente: No.*

Asistente: Entonces ¿qué desea hacer ahora?

En el ejemplo de la conversación A, el cliente no puede asegurar que el asistente virtual haya entendido que le ha dicho que no porque este agente conversacional no le ha corroborado la recepción de la información, ya sea de forma explícita o implícita. Entonces, se puede afirmar que en la conversación A no se ha establecido un punto de coincidencia entre la persona y el agente conversacional.

Sin embargo, en la conversación B, por el mero hecho de que el asistente virtual haya utilizado el adverbio «entonces», se sobreentiende que el agente conversacional ha entendido que el cliente ha respondido que no y le está lanzando una nueva pregunta. En este caso, sí se ha establecido el punto de coincidencia entre la persona y el agente conversacional, tan necesario para el correcto funcionamiento del agente.

Estructura de la conversación

Se ha comentado en los apartados anteriores que la conversación se compone de una serie de turnos de palabra relacionados entre sí y en los que los interlocutores exponen diferentes expresiones. También se ha explicado que la conversación es un acto colectivo en el que requiere que ambos interlocutores lleguen a establecer una serie de puntos de coincidencia. Para asegurar los fundamentos de la conversación y garantizar que el receptor del mensaje ha entendido al emisor, se han presentado algunas técnicas en las que se veía una clara correlación ente los actos de un turno de palabra y los del siguiente turno. De hecho, aunque las conversaciones tienen una estructura libre, se pueden identificar algunos patrones bien definidos en algunas partes de los diálogos entre humanos.

En numerosas ocasiones se va a dar el caso de que un turno de palabra esté relacionado con el siguiente. Por ejemplo, cuando se lanza una pregunta en un turno, se condiciona claramente a que en el siguiente turno de palabra se dé una respuesta.

Es por esta razón que se puede hablar de que en las conversaciones se da un patrón de **pares adyacentes** (Schegloff, 1968).

Los **pares adyacentes** son estructuras que se dan en una conversación del tipo pregunta-respuesta (pregunta seguida de una respuesta), propuesta-aceptación o propuesta-rechazo, petición-concesión, saludo-saludo….

Según el patrón de los pares adyacentes, en una conversación en la que se lance una propuesta (primera parte del par), acto seguido va a darse una aceptación o un rechazo (segunda parte del par). Sin embargo, se puede dar el caso de que en un diálogo, la segunda parte del par no aparezca directamente justo después de la primera parte, sino que haya algunas expresiones intermedias. Esta situación se podría dar si el interlocutor necesita hacer una pregunta aclaratoria para recabar información antes de poder aceptar o rechazar la propuesta que le había lanzado el otro interlocutor. En este caso se daría un subdiálogo entre la propuesta (primera parte del par) y la aceptación o rechazo de la misma (segunda parte del par).

Aparte de la estructura de la conversación marcada por los pares adyacentes, se pueden encontrar otros patrones en cómo se organiza una conversación. Por ejemplo, en una llamada de teléfono siempre habla primero la persona que responde al teléfono, luego lo hace la persona que llama indicándole quién es; en los pasos siguientes muy posiblemente se intercambien saludos y luego, la persona que llama indica el motivo de la llamada. Este patrón da comienzo a una llamada telefónica y es solo un ejemplo concreto, pero podríamos tener también un patrón para el final de la llamada. Además, en otros tipos de conversaciones podríamos encontrar otros patrones que describan la estructura de la conversación.

Los agentes conversacionales se van a beneficiar de los patrones en la estructura de la conversación para facilitar la generación de diálogos de forma automática. Por ejemplo, saber que la persona ha lanzado una pregunta y que, por lo tanto, el agente debe proporcionar una respuesta será una información muy útil a lo hora de crear un diálogo. Además, si se sabe que se está desarrollando un agente que responde a

llamadas telefónicas, se podrá prefijar el patrón que de comienzo o que finalice la conversación.

Información implícita en la conversación

En los apartados anteriores se han tratado las características de la conversación relacionadas con lo que se podría llamar la «infraestructura» de la conversación:

- La conversación es una actividad que realizan de forma conjunta varios interlocutores.
- Estos generan turnos de palabra en los que se exponen una serie de contribuciones que requieren la aceptación por parte del interlocutor y que cumplen con unos patrones predeterminados.

Sin embargo, hasta este punto no se ha comentado nada acerca de la información que se comunica entre los interlocutores durante el diálogo.

En una conversación, el significado de una contribución no se limita al significado de las palabras entendidas de forma individual. Es más, este significado tampoco se limita al significado que pueda tener una expresión concreta. Esto se debe a que la interpretación de una expresión se basa en algo más que el significado literal de las oraciones que la componen.

En la interpretación de una expresión intervienen el conocimiento que comparten los interlocutores, el contexto en el que se desarrolla la conversación o la intención del emisor del mensaje. Es por eso por lo que se puede decir que en una conversación hay información que tiene un significado literal o explícito, pero también hay otra información que el receptor del mensaje es capaz de inferir, por lo tanto, información implícita.

Las **implicaturas** o **informaciones implícitas** son los significados que no aparecen de forma explícita en la conversación, pero que el receptor del mensaje infiere.

La **inferencia** juega un papel crucial en una conversación, el emisor de un mensaje puede intentar hacer llegar al receptor más información de la que le transmite de forma explícita en las expresiones que componen su mensaje. Por ejemplo, si para hacer una reserva de un vuelo, un agente de viajes le pregunta a su cliente que qué día quiere viajar y este le responde que tiene que llegar para una reunión el 15 de mayo, entonces, el agente de viajes va a interpretar que tiene que reservar el vuelo cuya llegada se dé el día 14 de mayo. En este caso se ve cómo claramente el cliente no ha respondido al agente de viajes con la información que este le había preguntado, sino que le ha proporcionado otra información que de forma implícita responde a su pregunta. De hecho, en este caso el cliente ha decidido de forma intencionada proporcionarle al agente una información explícita que le permita a través de un proceso de inferencia obtener la información implícita requerida.

Los agentes conversacionales también deben ser capaces de extraer la información implícita que les hace llegar un humano cuando mantienen un diálogo. Para ello los agentes implementan mecanismos de inferencia que les permiten obtener los significados adicionales de un mensaje a partir del significado literal de las expresiones que lo componen.

Además, los agentes conversacionales para poder funcionar correctamente utilizando las informaciones implícitas deben ser capaces de determinar la intención de su interlocutor. El agente conversacional debe interpretar que el humano no se ha equivocado porque no haya respondido explícitamente la pregunta que le había formulado, sino que lo ha hecho a propósito porque la intención de la persona era transmitir al agente una información implícita.

La inferencia de información implícita en la conversación es una de las tareas más complicadas a las que se enfrenta un agente conversacional a la hora de comportarse

como una persona y de reproducir las características que tiene una conversación entre humanos.

Accede a los ejercicios de autoevaluación a través del aula virtual

10.4. Tipos de agentes conversacionales

Accede al vídeo «Agentes conversacionales» a través del aula virtual

Existen dos tipos de agentes conversacionales: los agentes conversacionales dedicados a una tarea concreta y los *chatbots*. Esta clasificación se hace con base en el tipo de conversación que son capaces de mantener los agentes conversacionales y la funcionalidad u objetivo final que pretenden implementar.

Agentes conversacionales dedicados

Los agentes conversacionales dedicados a una tarea concreta están diseñados para desarrollar una tarea en particular y están restringidos a un dominio de aplicación. Esta única tarea que puede realizar este tipo de agentes sería, por ejemplo, obtener algún tipo de información del usuario o ayudarle a completar una acción sencilla.

Los agentes conversacionales dedicados están configurados para mantener conversaciones cortas con la persona, en general conversaciones que involucren desde una interacción a unas pocas de interacciones (en el rango de media docena). Además, cabe resaltar que las conversaciones mantenidas por estos agentes son sencillas y no se realizan de una manera natural como cuando conversan dos humanos.

Los asistentes personales que encontramos en los teléfonos móviles o en algunos dispositivos inteligentes para el control domótico (Google Assistant, Siri de Apple, Cortana de Windows o Alexa de Amazon) son agentes conversacionales que forman parte de la categoría de agentes conversacionales dedicados a una tarea concreta. Estos pueden realizar tareas simples como, por ejemplo, dar instrucciones sobre un viaje, controlar electrodomésticos, encontrar restaurantes, ayudar a hacer llamadas telefónicas o enviar mensajes de texto.

Muchas empresas implementan agentes conversacionales dedicados en sus sitios web para ayudar a los clientes a responder a sus preguntas y dudas o para resolver los problemas que pudieran tener con los productos o servicios contratados. Además, los agentes conversacionales juegan un papel importante como interfaz hombre-máquina en algunos robots.

Chatbots

Los chatbots, llamados en español bots conversacionales, son aquellos agentes conversacionales que permiten entablar conversaciones extensas, es decir, con múltiples interacciones entre la persona y el agente conversacional.

Permiten mantener conversaciones no estructuradas, una característica fundamental de las conversaciones entre personas, y sobre cualquier tema o materia a diferencia de agentes conversacionales dedicados a una tarea concreta.

Por tanto, los *chatbots* ofrecen una conversación realista entre una persona y el agente conversacional utilizando el lenguaje natural.

> **Ejemplo ilustrativo 2.** Mitsuku - el chatbot ganador del Premio Loebner 2017.

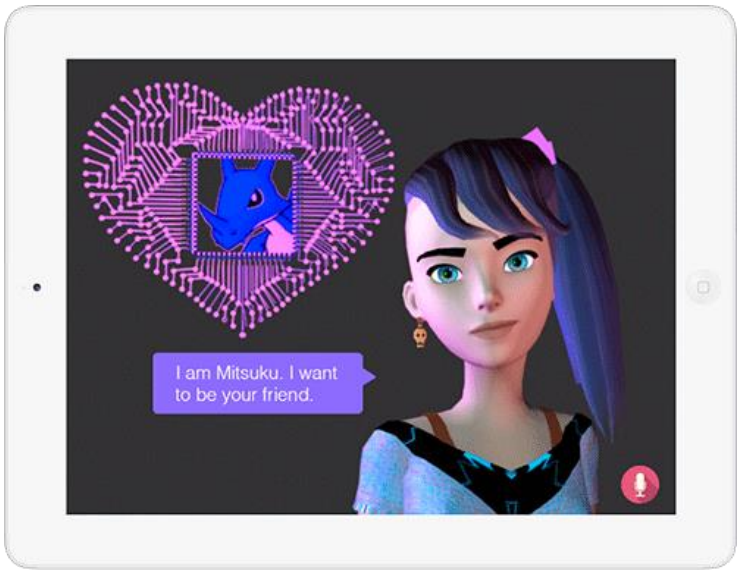


Figura 46. Avatar del *chatbot* Mitsuku.
Fuente: https://home.pandorabots.com/

Mitsuku está considerado como el mejor *chatbot* en la actualidad, ya que ha sido el ganador del Premio Loebner en las ediciones 2017, 2016 y 2013.

El **Premio Loebner** es una competición anual que premia a la aplicación de inteligencia artificial más inteligente del mundo. Para determinar el ganador de la competición lo que se hace es aplicar el test de Turing. A través de este test, un juez humano tiene que decidir si detrás de la pantalla a la que realiza preguntas está un programa de ordenador o un ser humano. Mitsuku consiguió superar el test de Turing gracias a que respondió de forma meditada y a un ritmo razonable y, además, contestó las preguntas de forma convincente.

Puedes observar el funcionamiento de Mitsuku, en el siguiente enlace: https://www.pandorabots.com/mitsuku/

Aunque en la prensa y la industria se utilice la palabra *chatbot* para nombrar cualquier agente conversacional, en este curso solo se va a utilizar este término para designar al grupo de agentes conversacionales que pueden mantener una conversación prolongada y realista sobre cualquier tema, tal como se hace en la comunidad del procesamiento del lenguaje natural. Por lo tanto, Mitsuku es considerado un *chatbot,* pero no Google Assistant, aunque sea capaz de desarrollar llamadas de forma fluida

según las últimas funcionalidades presentadas en el congreso de desarrolladores Google I/O que tuvo lugar en mayo de 2018.

Accede a los ejercicios de autoevaluación a través del aula virtual

10.5. Estructura de los agentes conversacionales

Accede al vídeo «Agentes conversacionales» a través del aula virtual

Los agentes conversacionales, aunque puedan utilizar diferentes modalidades para interactuar con el usuario e implementen diferentes métodos para gestionar el diálogo, siguen una estructura básica común.

La figura 2 muestra la arquitectura de un agente conversacional basado en voz:

- Un módulo de entrada para el reconocimiento automático de la voz.
- Un módulo de comprensión del lenguaje natural.
- Un módulo de gestión del diálogo.
- Un módulo de generación del lenguaje natural.
- Un módulo de una salida de conversión de texto al habla.

La arquitectura para un agente conversacional basado en texto sería equivalente, aunque el módulo de entrada implementaría el reconocimiento automático de texto y el módulo de salida renderizaría el texto a un formato de presentación gráfica adecuado.

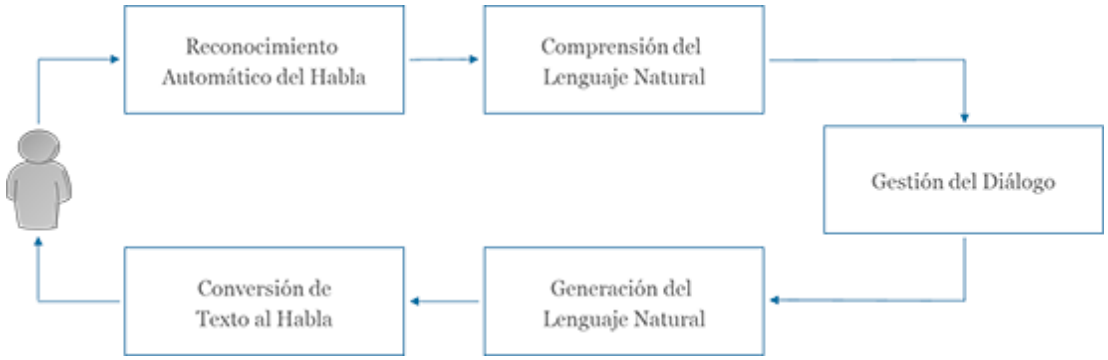


Figura 47. Arquitectura de un agente conversacional.

Reconocimiento automático del habla

El módulo de entrada tiene como función el reconocimiento automático del habla, es decir, extraer las **palabras a partir de la señal de audio** que se ha grabado con un micrófono y que recoge la voz del usuario. Aquí se utilizan tecnologías estándar para el modelado acústico-fonético, sin embargo, otros aspectos se pueden optimizar para hacer que su implementación sea más adecuada a los agentes conversacionales. Los métodos utilizados para el reconocimiento del habla no son objeto de estudio en este curso, aunque en los siguientes párrafos se describen brevemente algunas estrategias que permiten optimizar estos métodos.

Si el agente conversacional está dedicado a una tarea concreta y, por tanto, no se espera que vaya a responder a cualquier pregunta, solo aquellas que tienen que ver con la aplicación concreta, se puede hacer que solo sea capaz de transcribir frases en ese ámbito. Con este fin, los agentes conversacionales pueden utilizar modelos basados en **gramáticas de estados finitos** que especifican todas las respuestas posibles que el sistema entiende.

Otra opción para optimizar los modelos de lenguaje es que los agentes conversacionales utilicen un modelo de lenguaje N-gram en el cual las probabilidades en el diálogo de estados se adaptan para incorporar las restricciones concretas relacionadas con el contexto del diálogo.

Comprensión del lenguaje natural

El módulo de comprensión del lenguaje natural realiza las tareas básicas de procesamiento del lenguaje natural para extraer la **semántica** de las frases que recibe del módulo de reconocimiento automático del habla. Para representar la semántica se pueden utilizar diferentes alternativas y en el proceso de extracción se usan algunas de las técnicas de procesamiento del lenguaje natural que se han explicado en los capítulos anteriores.

En el caso de los agentes conversacionales dedicados, lo más común es utilizar *frames* y pares atributo-valor para representar la semántica de las frases. Entonces, estos agentes conversacionales basados en *frames* utilizan normalmente una ontología para la representación formal de la semántica de la conversación. Esta ontología define los diferentes *frames*, es decir, las posibles plantillas para una frase, una colección de los huecos (*slots*) en cada plantilla y los posibles valores para rellanar estos huecos.

Los pares atributo-valor restringen la tipología semántica de los valores que pueden ir asociados a cada hueco en la plantilla.

El módulo de comprensión del lenguaje natural de un agente conversacional basado en *frames* extrae la semántica de la petición lanzada por el usuario identificando diferentes elementos. De hecho, si el agente conversacional puede realizar tareas en múltiples dominios, primero **identifica el dominio** al que hace referencia el usuario. Por ejemplo, detecta si el usuario está hablando de reservar vuelos, programando la alarma de un reloj o gestionando las citas de su calendario. Evidentemente, en los agentes dedicados que trabajan en un único dominio no sería necesario realizar este paso. Sin embargo, la mayoría de los agentes conversacionales modernos tipo Siri de Apple, Alexa de Amazon o Google Assistant realizan varías tareas en algunos dominios predefinidos y necesitan implementar esta funcionalidad para clasificar la petición del usuario de acuerdo con el dominio al que hace referencia.

Una vez se ha identificado el dominio en el que se clasifica la petición lanzada por el usuario, lo siguiente es determinar la **intención del usuario**. Es decir, se tiene que identificar cuál es el objetivo del usuario y qué tarea pretende que realice el agente conversacional. Por ejemplo, en el dominio de la búsqueda de vuelos, la intención del usuario podría ser mostrar todos los vuelos entre dos ciudades para una fecha en concreto. En el otro ejemplo, el agente conversacional que gestiona la alarma de un reloj, las tareas que podría realizar serían activar la alarma a una hora concreta o desactivarla dependiendo de la intención del usuario.

Finalmente, cuando el agente conversacional ya ha detectado el dominio y la intención del usuario, a partir de la petición lanzada por el usuario también debe identificar los huecos (*slots*) en la plantilla que define la tarea que tiene que realizar el agente conversacional y obtener los valores necesarios para rellenar esos diferentes huecos de la plantilla.

El análisis semántico realizado por el módulo de comprensión del lenguaje natural requiere no solo extraer y representar el significado de la oración, incluyendo el dominio, intención y valores de los huecos del *frame*, sino también identificar otros aspectos propios de un diálogo como son los **puntos de coincidencia**. Entonces, en el proceso de extracción de la semántica de la petición lanzada por el usuario se deben identificar también los turnos de confirmación, negación o cortesía. Por ejemplo, si el módulo de comprensión del lenguaje natural recibe la frase «Sí, me gustaría saber el precio del vuelo de Madrid a Logroño», este debe identificar que la palabra «Sí» al inicio de la frase se refiere a una afirmación y es independiente de la tarea que debe realizar el agente conversacional, que es la consulta del precio de un vuelo concreto.

> **Ejemplo ilustrativo 3.** Representación semántica utilizando *frames* de una interacción con en un agente conversacional dedicado a la reserva de vuelos.
>
> El agente conversacional basado en *frames* define con una ontología los pares atributo-valor, es decir los huecos (*slots*) en la plantilla y los posibles valores para rellanar estos huecos. La

tabla 1 muestra algunos ejemplos de pares atributo-valor definidos para el agente conversacional que reserva vuelos.

Hueco (slot)	Valor (type)
ORIGIN_CITY	city
DESTINATION_CITY	city
DEPARTURE_DATE	date
ARRIVAL_ DATE	date
DEPARTURE_TIME	time
ARRIVAL_TIME	time

Tabla 9. Pares atributo-valor definidos para el agente conversacional que reserva vuelos.

Se observa en la tabla 1 que el hueco llamado `ORIGIN_CITY` y el llamado `DESTINATION_CITY` deben tener un valor que sea una ciudad (`city`). Por tanto, si aparecen esos huecos en la plantilla, solo se pueden rellenar con nombres de ciudades.

Además, la ontología que define los pares atributo-valor para los huecos y sus valores puede proporcionar una estructura jerárquica. Así, el valor date para los huecos `DEPARTURE_DATE` y ARRIVAL_ DATE de la tabla 1 puede ser a su vez el siguiente *frame* con valores del tipo entero (`INTEGER`), nominal (`NAME`) o miembros de un conjunto de nombres (`MEMBER`):

```
DATE
 MONTH NAME
 DAY (BOUNDED-INTEGER 1 31)
 YEAR INTEGER
 WEEKDAY (MEMBER (SUNDAY MONDAY TUESDAY WEDNESDAY
THURSDAY FRIDAY SATURDAY))
```

Cuando en un turno de palabra el módulo de comprensión del lenguaje natural recibe la petición en inglés «*Show me flights from Boston to San Francisco on Tuesday morning*», lo primero que hace es identificar el dominio al que hace referencia el usuario: el de los viajes en avión (`AIR-TRAVEL`). Además, este módulo identifica que la intención del usuario es ver los vuelos

disponibles y que, por lo tanto, la tarea que debe realizar es mostrar los posibles vuelos (`SHOW-FLIGHTS`).

El módulo de comprensión del lenguaje natural es también capaz de identificar varios huecos y obtener sus valores a partir de la petición lanzada por el usuario. Este módulo extrae la información que el origen del vuelo es Boston y el destino es San Francisco, y determina que la fecha del vuelo es el martes (*Tuesday*) por la mañana (*morning*).

Entonces, el módulo de comprensión del lenguaje natural basado en *frames* genera la siguiente representación semántica de la petición en inglés «*Show me flights from Boston to San Francisco on Tuesday morning*»:

```
DOMAIN: AIR-TRAVEL
INTENT: SHOW-FLIGHTS
ORIGIN_CITY: Boston
DEPARTURE_DATE: Tuesday
DEPARTURE_TIME: morning
DESTINATION_CITY: San Francisco
```

El módulo de comprensión del lenguaje natural de un agente conversacional basado en *frames* extrae la semántica de la petición lanzada por el usuario aplicando algunas de las técnicas de procesamiento del lenguaje natural que se han explicado en los capítulos anteriores. Por ejemplo, algunos agentes conversacionales como el Core Language Engine utilizan gramáticas de unificación con la semántica adjunta para identificar el significado de la frase y, a partir de esta, se extraen las posibles palabras que pueden completar los huecos en el *frame*.

Muchos de los agentes conversacionales comerciales en la actualidad y también el clásico agente conversacional para la reserva de viajes GUS (Bobrow et al., 1977) utilizan analizadores semánticos basados en **gramáticas semánticas**. Estas pertenecen al grupo de las gramáticas libres de contexto y tienen como característica que en la parte izquierda de las reglas que componen la gramática aparecen los nombres de los posibles huecos en las plantillas (*frames*).

Ventajas

La **ventaja** de las soluciones basadas en gramáticas semánticas es que el análisis semántico se puede implementar utilizando cualquier algoritmo estándar que permita realizar el análisis basado en una gramática libre de contexto o basado en programación dinámica como, por ejemplo, CKY. El resultado del analizador es la cadena de caracteres de entrada etiquetada jerárquicamente con nodos semánticos. Muchas soluciones para la comprensión del lenguaje natural utilizan gramáticas semánticas porque algunos nodos obtenidos del análisis se corresponden directamente con los huecos en un *frame*.

Ejemplo ilustrativo 4. Gramática semántica utilizada por el módulo de comprensión del lenguaje natural para la identificación de la intención y la obtención de los huecos y sus valores a partir de una petición lanzada por el usuario.

Para obtener la representación semántica de la petición en inglés «*Show me flights from Boston to San Francisco on Tuesday morning*» presentada en el ejemplo ilustrativo 3, es necesario que el módulo de comprensión del lenguaje natural basado en *frames* trabaje con una gramática semántica.

A continuación, se presenta un fragmento de una gramática semántica que permite realizar el análisis semántico de la petición del usuario que quiere ver distintos vuelos:

```
SHOW → show me | i want | can i see | …
DEPART_TIME_RANGE → (after | around | before)
                    HOUR |
                    morning | afternoon | evening
HOUR → one | two | three | four … | twelve
                    (AMPM)
FLIGHTS → (a) flight | flights
AMPM → am | pm
ORIGIN → from CITY
DESTINATION → to CITY
CITY → Boston | San Francisco | Denver |
                    Washington
```

Entonces, utilizando cualquier analizador para gramáticas libres de contexto y que permita generar un árbol a partir de la gramática anterior, se obtendría la estructura jerárquica de las etiquetas semánticas resultantes de analizar la petición del usuario. Considerando la gramática presentada justo arriba y la petición «*Show me flights from Boston to San Francisco on Tuesday morning*», la figura 3 presenta el resultado del análisis semántico utilizando la gramática semántica.

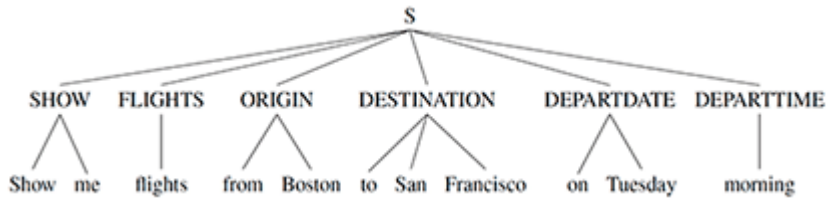


Figura 48. Resultado del análisis semántico utilizando la gramática semántica que permite crear un árbol donde los nodos son los posibles huecos de los *frames*. Fuente: Jurafsky y Martin, 2009.

Por ejemplo, en la figura 3 se observa que del nodo ORIGIN en el árbol se puede obtener el valor a asignar al hueco ORIGIN_CITY en el *frame* que representa la petición del Ejemplo ilustrativo 3. Además, la intención de la petición se puede obtener de la combinación de los nodos SHOW y FLIGHTS.

Problemas

Aunque muchas soluciones para la comprensión del lenguaje natural utilizan gramáticas semánticas debido a sus ventajas y su alta precisión para un dominio restringido, las gramáticas semánticas tienen varios problemas.

- El primero es que no funcionan cuando se dan ambigüedades; para solucionarlo, se pueden añadir probabilidades en la gramática semántica y utilizar un modelo basado en las gramáticas libres de contexto probabilísticas.

- Otro problema de las gramáticas semánticas es que estas se tienen que generar de forma manual, lo que es lento y costoso. Como alternativa existe la probabilidad de utilizar aprendizaje automático supervisado para aprender las

reglas, suponiendo que se disponga de un conjunto de entrenamiento donde cada oración está etiquetada con la semántica correcta, se puede entrenar un clasificador para que asigne a cada oración su dominio y sus intenciones, y generar un modelo que mapee de la frase a los valores de los diferentes huecos en el *frame*.

Una alternativa a las gramáticas semánticas que se podría implementar en el módulo de comprensión del lenguaje natural sería un modelo **HMM semántico** (Pieraccini y Levin, 1991); este método probabilístico evita que se tengan que codificar las gramáticas de forma manual. En este modelo HMM semántico los estados ocultos son las etiquetas de los huecos, mientras que las palabras observadas serían los valores con las que se podrían rellenar los huecos.

Se acaba de describir en detalle el funcionamiento del módulo de comprensión del lenguaje natural para agentes conversacionales basados en *frames*. Los agentes conversacionales comerciales actuales utilizan esa arquitectura, sin embargo, en el ámbito de la investigación se están desarrollando agentes más modernos que implementan una arquitectura basada en el diálogo o, mejor dicho, en las características del diálogo.

Este nuevo tipo de agentes conversacionales **basados en el diálogo** permiten, además de hacer preguntas, realizar otro tipo de acciones como informar de un hecho, sugerir algo o dar órdenes. Por lo tanto, estos agentes necesitan identificar los diferentes actos de habla que se dan en la conversación con el usuario. De hecho, la identificación de los actos de habla permite saber si el usuario simplemente está proporcionando una información al agente conversacional, si está afirmando o confirmando algo, haciendo una pregunta, etc.

El módulo de comprensión del lenguaje natural en un agente conversacional basado en el diálogo debe identificar los actos de habla, además de obtener los valores de los huecos del *frame* para representar el significado de la interacción.

Primero, se identifican los actos de habla aplicando técnicas de aprendizaje supervisado. Se entrena un clasificador a partir de un corpus etiquetado donde cada expresión se ha anotado a mano con su acto de habla. Para entrenar, se utiliza una gran variedad de características: características léxicas (bigramas; palabras como «no», que indican negación; o «muéstrame», que indica una petición; la longitud de la expresión o la puntuación) o características semánticas (contexto del diálogo o el acto de habla de la interacción previa).

Luego se obtienen los valores de los huecos, tal como se hacía en el caso de los agentes conversacionales basados en *frames*. En estos módulos se tienen en cuenta otros aspectos propios del diálogo como son los puntos de coincidencia. Entonces, en el proceso de extracción de la semántica de una interacción con el agente conversacional se identifican también los turnos de confirmación, negación o cortesía. Como hemos ejemplificado antes, si el módulo de comprensión del lenguaje natural recibe la frase «Sí, me gustaría saber el precio del vuelo de Madrid a Logroño», este debe identificar que la palabra «Sí» al inicio de la frase se refiere a una afirmación y que es independiente de la petición de la consulta del precio del vuelo.

Gestión del diálogo

El módulo de gestión del diálogo, también llamado controlador del dialogo, es el elemento principal de los agentes conversacionales y se encarga de identificar la acción que debe realizar el agente conversacional en el siguiente turno de palabra y cómo se debe continuar la conversación. Para ello este módulo analiza la representación semántica de la petición que ha efectuado el usuario del agente conversacional o de la frase que ha pronunciado la persona.

Así, analiza la representación obtenida por el módulo de comprensión del lenguaje natural y proporciona a su salida la acción a realizar o el concepto que se quiere transmitir al usuario del sistema conversacional en el siguiente turno de palabra y como respuesta a la petición realizada por el usuario.

La gestión del diálogo requiere mantener el estado y el flujo de la conversación y en algunos casos pedir más información al usuario para poder completar la información necesaria y dar una respuesta. Por ejemplo, si el usuario ha realizado la petición «Me gustaría saber el precio del vuelo de Madrid a Logroño», el agente conversacional solo sabrá a qué vuelo se refiere el usuario si mantiene el estado de la conversación y en alguna interacción anterior este le ha dado más detalles sobre el vuelo en cuestión, por ejemplo, el día en el que quiere viajar. Si ese no fuera el caso, el agente conversacional no puede responder directamente al usuario con el precio del vuelo y necesitaría realizarle una serie de preguntas para obtener la información que le falta sobre la fecha concreta del vuelo antes de poder ejecutar la búsqueda de los precios y contestar a la petición del usuario mostrándole los precios de los vuelos.

De esta forma, cualquier agente conversacional que necesite asegurar que puede responder a las peticiones del usuario, aunque estas sean incompletas, y que no solamente conteste a las peticiones para las cuales el usuario proporcione toda la información en una única petición, debe implementar un módulo de gestión del diálogo que mantenga el estado de la conversación y que permita múltiples interacciones con el usuario para obtener la información no disponible. De hecho, si el agente conversacional solo respondiera a las peticiones completas y no mantuviera ninguna interacción con el usuario, se hablaría de un simple sistema de preguntas y respuestas y no de un agente conversacional.

Existen diferentes estrategias para la gestión del diálogo. Por ejemplo, la mayoría de los agentes conversacionales dedicados que utilizan *frames* tienen un módulo de diálogo basado en una máquina de estados finitos que define las acciones necesarias para llevar a cabo una tarea concreta:

- La máquina de estados finitos que rige un agente conversacional dedicado basado en *frames* gestiona el dialogo y modela cómo este debe desarrollarse.
- Los estados de la máquina de estados finitos se corresponden con las preguntas a realizar al usuario y que son necesarias para obtener los valores de los diferentes huecos en la plantilla.

- Los arcos de la máquina de estados finitos se corresponden a las acciones que deben realizarse en función de lo que el usuario responda.

Entonces, la máquina de estados finitos controla completamente la conversación con el usuario. El agente conversacional le hace una serie de preguntas al usuario, ignorando (o malinterpretando) cualquier contestación que no sea una respuesta directa a la pregunta y luego pasa a la siguiente pregunta. De hecho, la arquitectura de control para un agente conversacional basado en *frames* también utiliza plantillas para generar el contenido de las nuevas oraciones con las que se debe continuar la conversación. Por lo que va a ser necesario rellenar los huecos en las plantillas con los valores relevantes que se han obtenido de las interacciones previas con el usuario.

Ejemplo ilustrativo 5. Gestión del diálogo en un agente conversacional dedicado basado en *frames* utilizando una máquina de estados finitos.

La máquina de estados finitos que rige un agente conversacional dedicado a la reserva de vuelos (similar al que se ha presentado en el ejemplo ilustrativo 3 y 4) se muestra en la figura 4. Este agente conversacional basado en *frames* gestiona el dialogo y modela cómo se debe desarrollar con base en esta máquina de estados finitos. De hecho, se observan las diferentes preguntas que se van a realizar para obtener la información necesaria para rellenar los diferentes huecos del *frame* antes de ejecutar la tarea de reservar el vuelo.

Por ejemplo, en uno de los turnos de palabra el agente conversacional pregunta «*Do you want to go from* `<FROM>` *to* `<TO>` *on* `<DATE>` *?*». Se observa que esta pregunta es una plantilla y que tiene tres huecos `FROM`, `TO` y `DATE` que deben ser rellenados a partir de las respuestas obtenidas de las preguntas realizadas en turnos anteriores.

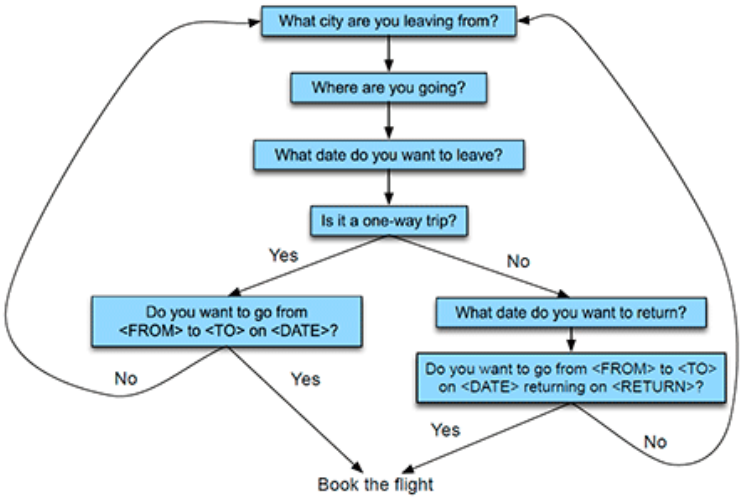


Figura 49. Máquina de estados finitos para modelar la gestión del diálogo de un agente conversacional basado en *frames* para reservar vuelos.
Fuente: Jurafsky y Martin, 2009.

La tabla 2 muestra la asociación entre los huecos y las preguntas a partir de las cuales se va a obtener el valor para rellenar esos huecos.

Hueco (slot)	Pregunta
FROM	What city are you leaving from?
TO	Where are you going?
DATE	What date do you want to leave?
RETURN	What date do you want to return?

Tabla 10. Asociación entre los huecos del *frame* y las preguntas que necesita realizar el agente conversacional para determinar el valor de esos huecos.

Una estrategia de gestión del diálogo basada en una máquina de estados finitos en la cual el agente conversacional toma totalmente la iniciativa tiene la ventaja de que el sistema siempre sabe a qué pregunta responde el usuario. Esto significa que el sistema puede preparar el módulo de reconocimiento automático del habla con un modelo de lenguaje adaptado a las respuestas para esta pregunta. Esto también facilita el funcionamiento del módulo de comprensión del lenguaje natural que puede obtener la semántica de una forma más fácil. No obstante, una arquitectura basada

en una máquina de estados finitos tan simplista solo se puede aplicar a tareas muy sencillas y no es suficiente para los agentes conversacionales que necesitan algo más de flexibilidad en el diálogo.

Google Assistant, Siri de Apple o Alexa de Amazon utilizan una estrategia de gestión del diálogo más compleja y que está basada en la arquitectura del agente conversacional GUS (Bobrow et al., 1977). Esta estrategia permite al usuario tomar cierta iniciativa y ayuda al agente conversacional a tratar los *frames* de una forma mucho más flexible. El agente conversacional hace preguntas al usuario, rellenando cualquier hueco en la plantilla al que haga referencia la información que el usuario le transmite. Entonces, en cada interacción, el agente conversacional puede rellenar varios huecos a la vez, incluso si la respuesta del usuario no se corresponde con la pregunta realizada. El sistema simplemente salta las preguntas asociadas con los huecos que ya han sido rellenados previamente.

Los diferentes huecos del *frame* se rellenan sin seguir una secuencia concreta y en función de la información proporcionada por el usuario.

Una vez que el agente conversacional tiene la suficiente información, ya puede realizar la acción, por ejemplo, consultar la base de datos de vuelos, y finalmente devolver el resultado al usuario.

Los agentes conversacionales basados en *frames* que siguen una estrategia de control flexible como, por ejemplo, muchos de los agentes conversacionales actuales, necesitan cambiar el control del diálogo y saltar entre *frames* dependiendo de la información proporcionada por el usuario. Debido a esta necesidad de cambiar dinámicamente el control, la arquitectura GUS para la gestión del diálogo se implementa como un sistema de reglas de producción y la información proporcionada por el usuario puede activar varias reglas de producción, el resultado de lanzarlas permite rellenar múltiples huecos en diferentes *frames*.

A diferencia de los agentes conversacionales dedicados basados en *frames*, los agentes conversacionales basados en el diálogo aplican estrategias mucho más avanzadas para la gestión del diálogo. Por ejemplo, para la gestión del diálogo, algunos de estos agentes conversacionales modernos, que todavía están en el ámbito de la investigación, aplican una estrategia basada en el contexto local y otros una estrategia basada en procesos de decisión de Markov y aprendizaje por refuerzo.

Los agentes conversacionales que se basan en el diálogo e implementan en el módulo de gestión del diálogo una estrategia basada en el contexto local lo que hacen es predecir, para un turno de palabra, cuál es la acción a realizar según el estado del dialogo, es decir, con base en las acciones que han tomado el agente y el usuario en los turnos de palabra anteriores. Para simplificar, se puede considerar que el estado del diálogo principalmente solo va a depender de la última acción que ha realizado el agente conversacional, la última acción que ha realizado el usuario y del estado actual del *frame*, es decir, del conjunto de los huecos rellenos y los valores que estos toman.

A partir de un corpus de conversaciones lo suficientemente grande se puede estimar la probabilidad de que el agente conversacional tome una acción dadas las acciones que han tomado el agente conversacional y el usuario en el turno de palabra anterior y el estado actual del *frame*. Entonces, el módulo de gestión del diálogo del agente conversacional va a seleccionar la acción que maximice la probabilidad para ese turno de palabra.

Esta estrategia de gestión del diálogo que decide qué acción debe realizar el agente conversacional tiene un problema: solo se fundamenta en el pasado del diálogo, ignorando completamente si la acción que toma el agente conversacional es probable que conduzca al resultado esperado o no (por ejemplo, a completar correctamente la reserva de un vuelo). De hecho, en el momento de planificar la acción aún falta mucho para poder saber si el resultado será exitoso. Es por eso que hoy en día se están estudiando nuevas estrategias para la gestión del diálogo asentadas en procesos de decisión de Markov y aprendizaje por refuerzo.

Generación del lenguaje natural

El módulo de generación del lenguaje natural tiene como objetivo elegir los conceptos que se quieren expresar al usuario y, además, planificar cómo expresarlos en palabras. Entonces, la función de este módulo se puede separar en dos etapas: qué decir y cómo decirlo.

Primera etapa

En la primera etapa del módulo de generación del lenguaje se realiza una tarea de **planificación del contenido**, es decir, se decide qué contenido se debe expresar al usuario en cada turno de palabra. Por ejemplo, si el usuario ha hecho una pregunta, se presenta la respuesta a esa pregunta o, si ha hecho una propuesta, el agente conversacional puede aceptarla o rechazarla. Así, la tarea de planificación del contenido está basada en el patrón de los pares adyacentes que define la estructura de las conversaciones y tiene en consideración los actos de habla asociados a cada una de las expresiones en la conversación.

Muchas veces la funcionalidad de planificación del contenido se integra con el módulo de gestión de diálogo y no aparece como un componente separado en el módulo de generación del lenguaje. Este sería el caso de los agentes conversacionales basados en *frames* que se han presentado en los ejemplos ilustrativos 3, 4 y 5. Estos agentes gestionan el diálogo utilizando una máquina de estados finitos de forma que, para cada estado, se obtiene directamente la expresión que el sistema le transmite al usuario en ese turno de palabra. De forma similar, los agentes conversacionales basados en el diálogo que aplican una estrategia según el contexto local también planifican la acción a realizar para un turno de palabra en el módulo de gestión de diálogo.

Segunda etapa

En la segunda etapa del módulo de generación del lenguaje se escogen las **estructuras sintácticas y las palabras** que se necesitan para expresar el concepto que se quiere transmitir al usuario. Existen diferentes formas para realizar esta tarea dependiendo de la estrategia utilizada para la gestión del diálogo en el módulo de gestión de diálogo o la implementación concreta de primera etapa del módulo de generación del lenguaje.

En el caso de un agente conversacional dedicado basado en *frames*, el módulo de gestión de diálogo ya ha determinado la pregunta que el agente conversacional debe efectuar al usuario en el siguiente turno de palabra. De hecho, esa pregunta que se transmitirá al usuario puede estar completa o tener algunos huecos que se deben rellenar con los valores obtenidos por el agente conversacional en las interacciones anteriores. Por lo tanto, el módulo de generación del lenguaje natural no tiene que realizar ninguna acción complicada, solo identificar los huecos en la frase y rellenarlos con los valores correctos para ese turno de palabra.

En el caso de un agente conversacional que se basa en el diálogo e implementa una estrategia de gestión basada en el contexto local, el módulo de gestión de diálogo ha determinado el acto de habla del siguiente turno de palabra, es decir, la acción que el agente conversacional debe realizar en el siguiente turno de palabra. Por lo tanto, el módulo de generación del lenguaje natural conoce el acto de habla del siguiente turno de palabra y el estado actual del *frame*, es decir, el conjunto de los huecos y los valores que estos toman. Entonces, el módulo de generación del lenguaje natural debe generar el mensaje que el agente conversacional va a transmitir al usuario en el siguiente turno de palabra.

Para generar el mensaje asociado a un acto de habla se utiliza un proceso con dos pasos:

- Primero se genera una frase deslexicalizada (una frase con huecos).

- Luego se rellenan los huecos con los valores adecuados que se obtienen del estado actual del *frame*.

Se sigue esta estrategia de dos pasos porque generar una frase deslexicalizada es más fácil que generar la frase final. Para obtener la frase deslexicalizada lo más normal es utilizar un N-grama entrenado a partir de las frases de un corpus que están etiquetadas con el acto de habla que se quiere expresar en el siguiente turno de palabra. Sin embargo, algún trabajo reciente, en lugar de utilizar N-gramas, utiliza modelos neuronales que aprenden cómo obtener la frase resultante a partir del siguiente turno de palabra y del estado actual del *frame* (Wen et al., 2015).

Conversión del texto al habla

El módulo de salida tiene como función la **conversión del texto al habla** sintetizando en formato de voz el mensaje a transmitir al usuario en el siguiente turno de palabra: Este módulo de conversión del texto al habla obtiene del módulo de generación del lenguaje natural el mensaje en formato de texto y sintetiza la onda de sonido para este mensaje. Los métodos utilizados para la conversión del texto al habla no son objeto de estudio en este curso, aunque describiremos brevemente algunas ideas de cómo se realiza la síntesis de la voz.

El módulo de conversión del texto al habla asigna anotaciones sobre el acento y la entonación de cada una de las palabras que componen el mensaje (Reiter y Dale, 2000) y luego las utiliza para sintetizar una onda de sonido. Los métodos de síntesis que más se han utilizado hasta la fecha son los basados en concatenación. En estos métodos se graban unas muestras del habla y se guardan en una base de datos, después se combinan para crear nuevas palabras y frases. Sin embargo, hay otros métodos más modernos fundamentados en hacer las transiciones más suaves o en la articulación.

Accede a los ejercicios de autoevaluación a través del aula virtual

10.6. Diseño de *chatbots*

Accede al vídeo «Chatbots» a través del aula virtual

Los *chatbots*, aquellos agentes conversacionales que permiten reproducir conversaciones extensas y no estructuradas y que se han utilizado en el ámbito del entretenimiento o para propósitos prácticos como probar teorías de la práctica psicológica, se dividen en dos clases en función de su implementación: sistemas basados en reglas y sistemas basados en corpus.

Chatbots basados en reglas

Entre estos se incluye uno de los primeros agentes conversacionales, ELIZA (Weizenbaum, 1966). ELIZA fue diseñado para simular a un psiquiatra cuyo método involucra hacer reflexionar al paciente devolviéndole sus propias declaraciones. Por ejemplo, si un paciente le dice al psiquiatra: «Fui a dar un paseo en bote», el psiquiatra le va a responder: «Háble me de los botes». En este caso no se asumirá que el psiquiatra no sabe qué es un bote, se asume que la respuesta tiene un objetivo puramente conversacional. Entonces, un *chatbot* basado en reglas como ELIZA implementa este tipo de conversación poco común en la cual el *chatbot* no conoce casi nada del mundo real y solo se dedica a reformular las palabras del usuario.

Un *chatbot* basado en reglas como ELIZA utiliza **reglas patrón → transformación**. Cada regla describe la transformación que se debe aplicar a la declaración hecha por el usuario para obtener la respuesta que le va a proporcionar el *chatbot*, siempre y cuando la declaración del usuario cumpla un cierto patrón. Cada regla está asociada a una **palabra clave** que puede ocurrir en la frase, además, a las palabras clave se les asigna un rango, las palabras clave más específicas tienen un rango más alto y las más generales tienen un rango más bajo.

En cada turno de palabra, el *chatbot* responde con la transformación del patrón asociado a la palabra clave que tiene un rango más alto de todas las palabras contenidas en la declaración hecha por el usuario. En el caso de que no hubiera ninguna palabra en la declaración del usuario que coincidiera con las palabras claves asociadas a las reglas, el *chatbot* daría una respuesta evasiva como «esto es muy interesante» o «por favor, sigue». También puede utilizar algún truco para intentar evitar estas respuestas evasivas, por ejemplo, cuando le llega una frase en la que la palabra clave más probable es el determinante «mi», se aplica una transformación con un patrón similar a «Antes has dicho que tu …» y el resultado se guarda en memoria para utilizarlo en el próximo turno de palabra en el que no coincida ninguna palabra clave. La figura 5 muestra la lógica del algoritmo implementado por ELIZA.

```
function ELIZA GENERATOR(user sentence) returns response

Find the word w in sentence that has the highest keyword rank
  if w exists
      Choose the highest ranked rule r for w that matches sentence
      response ← Apply the transform in r to sentence
      if w = 'my'
          future ← Apply a transformation from the 'memory' rule list to sentence
          Push future onto memory stack
  else (no keyword applies)
    either
      response ← Apply the transform for the NONE keyword to sentence
    or
      response ← Pop the top response from the memory stack
  return(response)
```

Figura 50. Lógica del algoritmo implementado por ELIZA, *chatbot* basado en reglas.
Fuente: Jurafsky y Martin, 2009.

El modelo de reglas patrón → transformación de ELIZA aún se utiliza hoy en día. Por ejemplo, un *chatbot* moderno como ALICE se basa en una versión actualizada del *chatbot* basado en reglas ELIZA.

> **Ejemplo ilustrativo 5.** ELIZA, uno de los primeros agentes conversacionales, un *chatbot* basado en reglas que simula a un psiquiatra cuyo método involucra hacer reflexionar al paciente contestándole con sus propias declaraciones.

A continuación, se presenta un fragmento de la conversación más famosa de ELIZA recogida en el artículo de Weizenbaum (1966). Las frases en minúscula son las que pronunció el paciente y las frases en mayúscula las contestaciones generadas por parte del *chatbot* que simulaba el psiquiatra:

> *Men are all alike.*
> *IN WHAT WAY?*
> *They're always bugging us about something or other.*
> *CAN YOU THINK OF A SPECIFIC EXAMPLE?*
> *Well my boyfriend made me come here.*
> *YOUR BOYFRIEND MADE YOU COME HERE*
> *He says I'm depressed much of the time.*
> *I AM SORRY TO HEAR YOU ARE DEPRESSED*
> *…*
> *WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?*
> *My father*
> *YOUR FATHER*
> *You are like my father in some ways.*
> *WHAT RESEMBLANCE DO YOU SEE?*
> *You are not very aggressive but I think you don't want me to notice that.*
> *WHAT MAKES YOU THINK I AM NOT AGGRESSIVE?*
> *You don't argue with me.*
> *WHY DO YOU THINK I DON'T ARGUE WITH YOU?*
> *You are afraid of me.*
> *DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU?*
> *My father is afraid of everybody.*
> *WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER?*
> *Bullies.*
> *DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME?*

Algunas de las reglas patrón → transformación que implementa ELIZA son las siguientes:

```
(0 YOU 0 ME) -> (WHAT MAKES YOU THINK I 3 YOU)
(I *) -> (You say you 2)
```

En la primera regla, cuando se da el patrón `0 YOU 0 ME`, es decir, algo seguido de la palabra `YOU`, otro algo y luego la palabra `ME`, entonces se transforma en la frase en `WHAT MAKES YOU THINK I 3 YOU`, donde el número tres debe substituirse por el valor del segundo cero en el patrón. Por lo tanto, a partir de la primera regla, la oración pronunciada por el usuario «*You hate me*» iría

seguida de la oración obtenida en la transformación «*WHAT MAKES YOU THINK I HATE YOU*».

Si el usuario pronuncia la siguiente frase: «*I know everybody laughed at me*», ELIZA le contestaría «*YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU*» después de aplicar la transformación recogida en la segunda regla.

Chatbots basados en corpus

Los *chatbots* basados en corpus aplican técnicas de **minería de datos** a grandes conjuntos de conversaciones entre humanos con el fin de extraer las posibles respuestas del *chatbot* al usuario y para no tener que generar reglas a mano.
Los principales corpus que permiten aprender de las conversaciones están basados en conversaciones en plataformas de chat, en Twitter o en diálogos de películas (Serban et al., 2017). De hecho, a veces estos *chatbots* también pueden minar conversaciones entre un humano y una máquina e incluso extraer oraciones de un texto que no sea un diálogo.

Al igual que los basados en reglas, la mayoría de *chatbots* basados en corpus tienden a utilizar poca información del contexto de la conversación para generar una respuesta y suelen generar la respuesta basándose solamente en el turno de palabra inmediatamente anterior. Entonces, los *chatbots* basados en corpus tienen cierta similitud con los sistemas de búsqueda de respuestas (*Question Answering*) que permiten la recuperación de información basada en el lenguaje natural.

Hay dos tipos de *chatbots* basados en corpus:

- Basados en recuperación de información.
- Los *chatbots* secuencia a secuencia.

Por ejemplo, Cleverbot, que permite que un usuario pueda hablar con él por puro divertimento, pertenece al grupo de los *chatbots* basados en recuperación de información.

***Chatbots* basados en recuperación de información**

Este tipo de *chatbots* funcionan utilizando un método que les permite obtener una respuesta proporcionada por un humano en una conversación previa y utilizarla para responder al turno de palabra actual.

Así, usan una respuesta que aparece en el corpus y cuyo turno anterior es lo más parecido posible a la interacción que justo acaba de efectuar el usuario con el *chatbot* para el turno de respuesta de esta interacción. La idea es que se debe buscar un turno que se parezca lo más posible al turno del usuario y devolver la respuesta humana a ese turno (Jafarpour, Burges y Ritter, 2009; Leuski y Traum, 2011).

Otra opción sería que el *chatbot* basado en recuperación de información, en lugar de devolver la respuesta al turno más similar, devolviera directamente el turno más similar. La idea en este caso es buscar las coincidencias entre la interacción del usuario y los turnos en el corpus, ya que una buena respuesta a menudo compartirá palabras o semántica con el turno anterior.

Aunque devolver la respuesta al turno más similar parece ser un algoritmo más intuitivo, en la práctica parece que devolver el turno más similar funciona mejor (Ritter, Cherry y Dolan, 2011; Wang et al. 2013).

Para medir la similitud entre turnos se puede utilizar cualquier medida de similitud, por ejemplo, la similitud del coseno computada sobre las palabras (usando una medida de *term frequency–inverse document frequency*). De hecho, se puede extender el algoritmo para que incluya más características, además de las palabras en el turno actual, por ejemplo, las palabras en turnos anteriores o información sobre el usuario.

Debido a la importancia que tiene el corpus para el correcto funcionamiento de los *chatbots* basados en recuperación de información, las respuestas que va generando el usuario durante su interacción con el *chatbot* también se pueden usar para seguir entrenando el *chatbot*.

***Chatbots* secuencia a secuencia**

Estos *chatbots* están basados en el paradigma de la traducción automática y aplican técnicas de aprendizaje automático supervisado. Una forma alternativa de utilizar un corpus para generar un diálogo es pensar en la generación de respuestas como la tarea de transducción desde el turno del usuario hasta el siguiente turno del sistema, es decir, un chatbot que implemente esta idea sería básicamente una versión de ELIZA, pero basado en aprendizaje automático.

En la primera implementación de esta idea se utilizaron técnicas de traducción automática basada en frases (Ritter et al., 2011) para traducir un turno de usuario en una respuesta del sistema. Sin embargo, rápidamente quedó claro que la tarea de generación de respuestas era demasiado diferente de la traducción automática. En esta, las palabras o frases en las oraciones de origen y de destino se alinean entre sí. No obstante, en una conversación, una declaración del usuario puede no compartir palabras o frases con una respuesta coherente. Entonces, aproximadamente de forma simultánea se empezaron a modelar los transductores para la generación de respuestas utilizando modelos secuencia a secuencia (seq2seq) (Shang, Lu y Li, 2015; Vinyals y Le, 2015; Sordoni et al. 2015).

El **modelo seq2seq** tiene una serie de problemas, por ejemplo, es incapaz de modelar el contexto anterior a la conversación y genera respuestas que no tienen por qué ser coherentes entre los diferentes turnos de palabra. Entonces, se han realizado algunas modificaciones al modelo seq2seq básico para que pueda funcionar en *chatbots* secuencia a secuencia. Por ejemplo, se ha introducido información sobre los turnos de palabra previos utilizando un modelo jerárquico (Lowe et al., 2017) y se han

implementado métodos de aprendizaje por refuerzo para aprender a escoger las respuestas que hacen que la conversación sea más natural (Li et al., 2017).

Accede a los ejercicios de autoevaluación a través del aula virtual

10.7. Referencias bibliográficas

Austin, J. L. (1962). *How to Do Things with Words*. Cambridge: Harvard University Press.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H. y Winograd, T. (1977). GUS, A frame driven dialog system. *Artificial Intelligence*, 8, 155–173.

Clark, H. H. (1996). *Using Language*. Cambridge: Cambridge University Press.

Clark, H. H. y Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13, 259-294.

Forbes-Riley, K. y Litman, D. J. (2011). Benefits and challenges of real-time uncertainty detection and adaptation in a spoken dialogue computer tutor. *Speech Communication, 53*(9), 1115–1136.

Jafarpour, S., Burges, C. y Ritter, A. (2009). Filter, rank, and transfer the knowledge: Learning to chat. Recuperado de https://aritter.github.io/chat.pdf

Jurafsky, D. y Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computational Linguistics*. New Jersey, Estados Unidos: Prentice-Hall.

Leuski, A. y Traum, D. (2011). NPCEditor: Creatingvirtual human dialogue using information retrieval techniques. *AI Magazine, 32*(2), 42-56.

Levinson, S. C. (1983). Conversational Analysis. En *Pragmatics*. Cambridge: Cambridge University Press.

Li, J., Monroe, W., Shi, T., Ritter, A. y Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. En *EMNLP* (pp. 2157-2169). Copenhagen: Association for Computational Linguistics.

Lowe, R. T., Pow, N., Serban, I. V., Charlin, L., Liu, C. W. y Pineau, J. (2017). Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse, 8*(1), 31–65.

Pieraccini, R. y Levin, E. (1992). Stochastic representation of semantic structure for speech understanding. *Speech Communication, 11*(2-3), 283-288.

RAE. (S. f.). Diálogo. En Diccionario de la lengua española (actualización de la 23ª ed.). Recuperado de https://dle.rae.es/?id=DetWqMJ

Reiter, E. y Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press.

Ritter, A., Cherry, C. y Dolan, B. (2011). Data-driven response generation in social media. In EMNLP-11, pp. 583-593.

Sacks, H., Schegloff, E. A. y Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language, 50*(4), 696–735.

Schegloff, E. A. (1968). Sequencing in conversational openings. *American Anthropologist*, 70, 1075–1095.

Serban, I. V., Lowe, R. T., Charlin, L. y Pineau, J. (2017). A survey of available corpora for building data-driven dialogue systems. arXiv:1512.05742.

Shang, L., Lu, Z. y Li, H. (2015). Neural responding machine for short-text conversation. En *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (pp. 1577-1586). Beijing: Association for Computational Linguistic.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J. Y., Gao, J. y Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. Recuperado de http://rali.iro.umontreal.ca/rali/sites/default/files/publis/1506.06714.pdf

Stalnaker, R. C. (1978). Assertion. En P. Cole (Ed.), *Pragmatics: Syntax and Semantics* (Vol. 9, pp. 315-332). New York: Academic Press.

Vinyals, O. y Le, Q. (2015). A neural conversational model. En *Proceedings of the International Conference on Machine Learning*. Recuperado de https://arxiv.org/pdf/1506.05869.pdf

Wang, H., Lu, Z., Li, H. y Chen, E. (2013). A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 935-945). Seattle: Association for Computational Linguistics.

Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communication between man and machine. *Communications of the ACM, 9*(1), 36-45.

Wen, T. H., Gasic, M., Kim, D., Mrksic, N., Su, P. H., Vandyke, D. y Young, S. J. (2015). Stochastic language generation in dialogue using recurrent neural networks with

convolutional sentence reranking. En *SIGDIAL 2015 Meeting on Discourse and Dialogue* (pp. 275-284). Praga: Association for Computational Linguistics.

Yankelovich, N., Levow, G. A. y Marx, M. (1995). Designing SpeechActs: issues in speech user interfaces. En *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)* (pp. 369-376). New York: ACM Press/Addison-Wesley Publishing Co.

Accede al vídeo «Resumen» a través del aula virtual