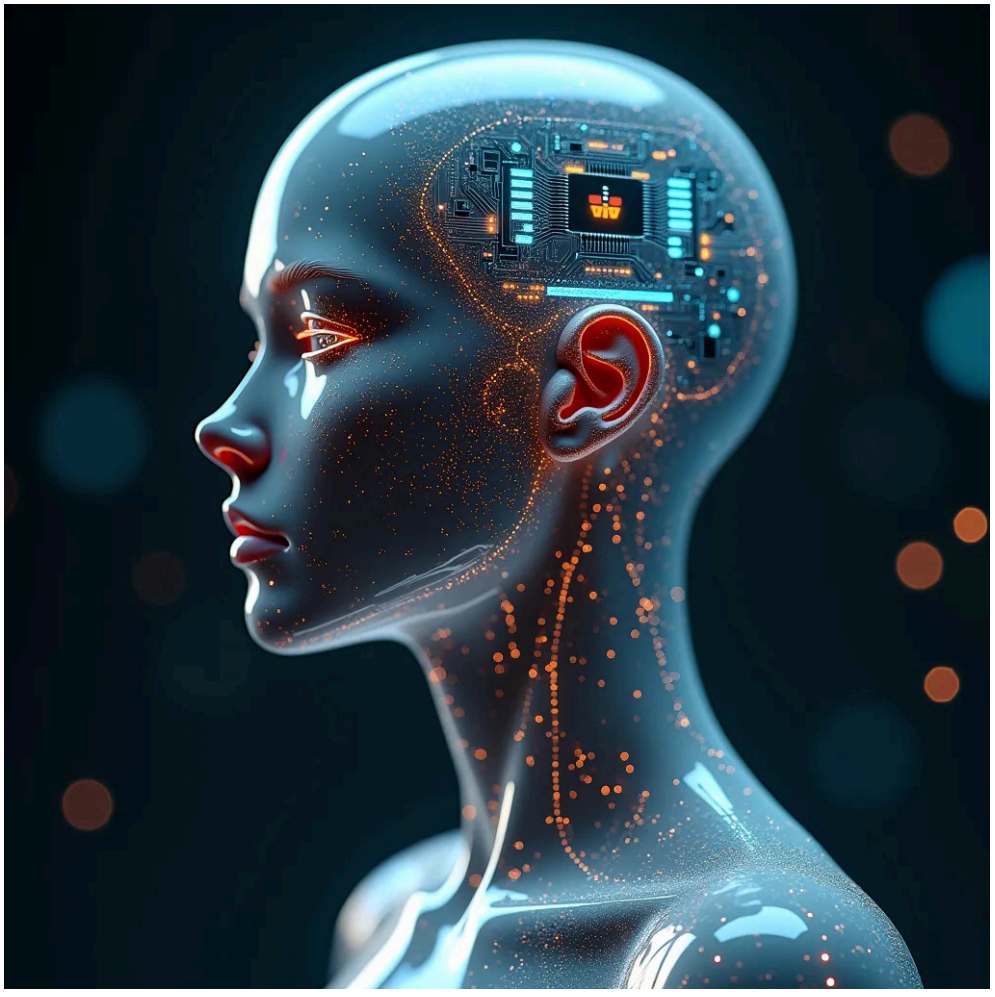
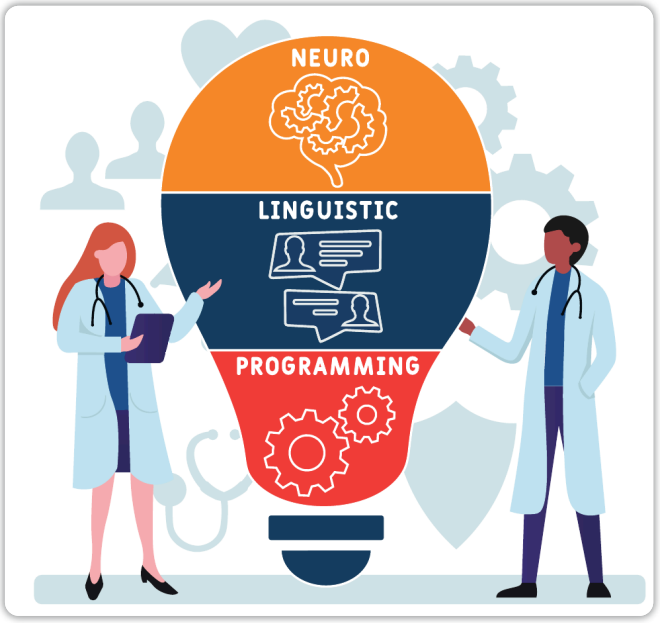
UD03_01: Natural Language Processing



1. Introducción

El procesamiento del lenguaje natural (PLN) o Natrual language Processing (NLP) está en el centro de una desconcertante pregunta: ¿cómo pueden dos entidades radicalmente diferentes –humanos y computadoras– comunicarse verdaderamente entre sí? El lenguaje humano es el producto complejo e imperfecto de la evolución social y biológica. Está lleno de excepciones ilógicas, matices sutiles y múltiples niveles de pensamiento abstracto. En contraste, las computadoras se comunican mediante modelos matemáticos que, por complejos que sean, siguen un conjunto de reglas lógicas y verificables. A medida que los sistemas digitales asumen un papel cada vez mayor en la actividad humana, deben ser capaces de interpretar correctamente lo que los humanos realmente quieren decir con las palabras que expresan.

El procesamiento del lenguaje natural es un campo muy amplio, que abarca diversas áreas, y relaciona varios campos de la técnica y la lingüística. Siguiendo la línea del test de Turing, en 1954 se llevó a cabo el **test de Georgetown** que supuso la traducción automática de unas 60 oraciones del ruso al inglés de manera exitosa.



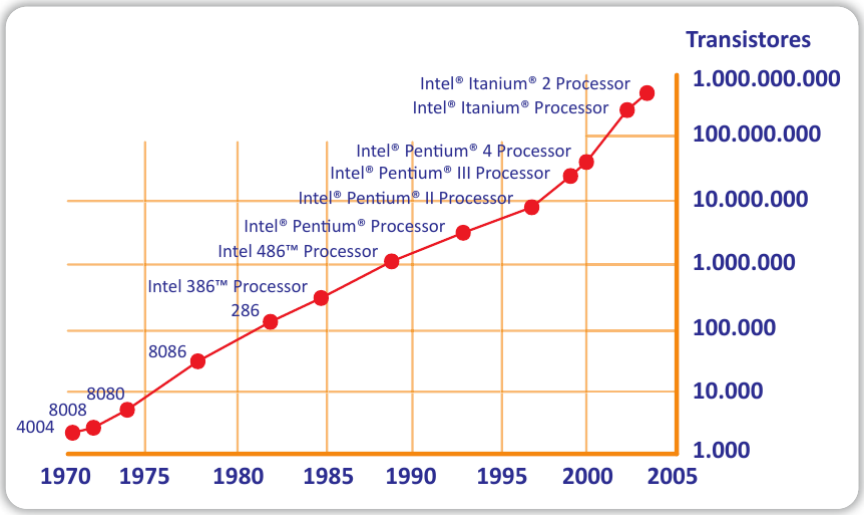
Definición El procesamiento del lenguaje natural estudia las relaciones del lenguaje entre los seres humanos y las máquinas.

Tras ello hubo un parón en el desarrollo de esta disciplina, debido a limitaciones del hardware, que perduraría hasta la década de 1980.

La ley de Moore indica que cada 2 años, desde 1970, se duplica el número de transistores en un microprocesador, lo que directamente expresa un aumento en términos de potencia de computación. Ese incremento de potencia, al igual que en otras disciplinas de la inteligencia artificial, como la visión artificial, ha sido lo que ha posibilitado el estado alcanzado

actualmente por la técnica.

El procesamiento del lenguaje natural implica necesariamente de la unión de dos áreas: el área asociada a la máquina y el área relacionada con el ser humano. Esta última comporta la forma actual de comunicación entre seres humanos, ya que, si el objetivo es tratar de que una máquina se comunique con un ser humano, y viceversa, es necesario el estudio formal de la manera en la que se comunica una persona, y es precisamente en este punto donde entra en juego la lingüística como disciplina de estudio.



Ampliación A largo de la historia de la humanidad ha habido distintos tipos de teorías lingüísticas comunes a varios idiomas, siendo muy común la de Noam Chomsky (por ejemplo, la gramática de constituyentes/transformacional).

Las gramáticas de Chomsky son un marco teórico para analizar lenguajes formales, clasificándolos en cuatro niveles según su poder expresivo y las restricciones de las reglas de producción. Esta clasificación se llama la **Jerarquía de Chomsky**.

1. **Gramática Tipo 0: Gramáticas no restringidas**

- **Definición**: Estas gramáticas son las más generales. Las reglas de producción pueden tener cualquier forma, siempre que la parte izquierda no esté vacía.
- **Forma de regla**: $\alpha \rightarrow \beta$, donde α y β son cadenas de símbolos, y $\alpha \neq \varepsilon$ (no puede ser vacío).
- **Ejemplo**:
    - Regla: $AB \rightarrow BA$
    - Entrada: AB
    - Resultado: BA
- **Uso:**

- **Procesamiento de lenguajes formales complejos:** Modelado de problemas matemáticos donde se requiere computación universal, como problemas de decisión que implican lógica de predicados.

2. **Gramática Tipo 1: Gramáticas sensibles al contexto**

- **Definición**: Las reglas deben mantener o aumentar la longitud de la cadena. Las producciones están restringidas a contextos específicos.
- **Forma de regla**: $\alpha A \beta \rightarrow \alpha \gamma \beta$, donde A es un símbolo no terminal y γ no es más corta que A.
- **Ejemplo**:
  - Regla: $A\ B \rightarrow A\ C\ B$
  - Entrada: $xABx$
  - Resultado: $xACBx$
- **Uso:**
  - **Verificación de gramática en lenguajes naturales:** Reglas como concordancia de género y número en oraciones humanas.
  - **Lenguajes de programación complejos:** Como los lenguajes de programación modernos, que requiere manejo contextual, por ejemplo, estructuras anidadas que dependen de tamaños específicos de datos.

3. **Gramática Tipo 2: Gramáticas libres de contexto**

- **Definición**: Las reglas tienen un lado izquierdo compuesto por un único no terminal.
- **Forma de regla**: $A \rightarrow \gamma$, donde A es un símbolo no terminal y γ es una cadena de símbolos.
- **Ejemplo**:
  - Reglas:
    - $S \rightarrow aSb$
    - $S \rightarrow \varepsilon$
  - Generación: $\varepsilon, ab, aabb, aaabbb$
- **Uso:**
  - **Parsers sintácticos:** Analizan estructuras como expresiones aritméticas, estructuras condicionales y bucles (por ejemplo, árboles de derivación en lenguajes como Java o Python).
  - Regla típica: $\text{expr} \rightarrow \text{expr} + \text{term} \mid \text{term}$

4. **Gramática Tipo 3: Gramáticas regulares**

- **Definición**: Las reglas de producción tienen una forma muy restringida, adecuada para describir lenguajes regulares.
- **Forma de regla**: $A \to aB$ o $A \to a$, donde A y B son no terminales y a es un terminal.
- **Ejemplo**:
    - Reglas:
        - $S \to aA$
        - $A \to bS$
        - $A \to b$
    - Generación: $ab, abab, ababab$
- **Uso:**
    - **Sistemas de búsqueda y validación:** Validar direcciones de correo electrónico, números de teléfono o patrones simples de texto.

Cada tipo de gramática tiene aplicaciones específicas en ciencias computacionales y lingüísticas, como compiladores y lenguajes naturales.

Resumen de aplicaciones:

Tipo de Gramática	Aplicación
Tipo 0	Problemas de decisión computacionales complejos.
Tipo 1	Lenguajes de programación y análisis contextual.
Tipo 2	Parsers y compiladores para lenguajes de programación.
Tipo 3	Validación de datos y procesamiento de cadenas simples.

Por tanto, la unión de **una falta de potencia de cálculo y del tipo de lingüística predominante hasta 1980** marcó el lento desarrollo del procesamiento del lenguaje natural.

A la hora de llevar a cabo avances en el campo del procesamiento del lenguaje natural, es necesario disponer de un set de datos (o corpus) específico para cada lengua y, por consiguiente, existirán modelos específicos para cada idioma; todo ello determina el papel del lingüista en un proyecto de inteligencia artificial.

Definición El objetivo general de los sistemas de Procesamiento del Lenguaje Natural (PLN) es el tratamiento de la lengua a fin de ser interpretada y/o producida a la manera en la que lo hacemos los seres humanos (COMPRENSIÓN). Es una tarea de gran complejidad.

Un **corpus** es un conjunto de palabras (un texto) de una lengua y se emplea para formar un diccionario, pero no en el sentido de documento donde se explica o definen palabras, sino como concepto de conjunto de palabras de una lengua. Diversos métodos de inteligencia artificial emplean corpus para entrenarse.

Ejemplo Si una persona española tuviera que pensar qué corpus emplear para el entrenamiento de un procesador de lenguaje natural en español, es fácil que centrara en las grandes obras literarias de nuestra lengua, como por ejemplo *El Quijote*, o un texto difundido en nuestra lengua, de amplia proyección, como puede ser *La Biblia*. Lo mismo ocurre en otros idiomas, y en inglés en concreto, existe una base de datos que contiene un conjunto de corpus en esta lengua, que fue realizado en 2014 y recibe el nombre de Gutemberg.

Ampliación En el siguiente link se puede ampliar información acerca de la base de datos Gutemberg: https://github.com/pgcorpus/gutenberg

En el link que aparece a continuación figura uno de los muchos corpus que pueden encontrarse para español: https://github.com/roquegv/spanishNLPModelCorpus

La gramática de constituyentes/transformacional en sus fundamentos teóricos no emplea predominantemente corpus, que es la llave en la que se basa el aprendizaje máquina para el procesamiento del lenguaje.

Tal como ya se ha estudiado, para que una máquina y un ser humano se comuniquen es preciso entender y estudiar la parte de la máquina y la parte del ser humano. Se iniciará este estudio comenzando por esta última, definiendo la lingüística y sus áreas principales.

Atendiendo a lo expuesto por D. Jurafsky en su texto «Speech and languaje processing», en el estudio de la lingüística de una lengua no solo es necesario llevar a cabo el estudio de las palabras y expresiones regulares del mismo, sino que es preciso estudiar las relaciones generadas por estos cuatro campos: **morfología**, **sintaxis**, **semántica** y **pragmática**.

La disciplina que hace uso de sistemas de computación para la compresión y la generación de lenguas naturales es la:

- Lingüística Computacional
- Ingeniería Lingüística

Un lingüista en un proyecto de inteligencia artificial aporta el conocimiento y el enfoque para llevar a cabo exitosamente la programación de las complejas estructuras (variables en virtud de los detalles de origen y evolución de cada lengua) entre los diferentes caracteres para formar una palabra, y de las diversas palabras para crear oraciones. Una clave de esta labor es el etiquetado, por el cual a una palabra, grafía, sintagma o estructura determinada se le asigna una etiqueta que lo clasifica.

> Ejemplo **Perro** es la combinación de las grafías «p» «e» «r» «r» «o», que a su vez desde un punto semántico representa el concepto de un animal, y cuya vocal «o» final denota el género de la palabra, que hace referencia al sexo del animal.

Dejando de lado conceptos técnicos más específicos, como el uso de N-gramas, expresiones regulares para la búsqueda de cadenas, transductores de estado finito, o modelos como el de cadenas ocultas de Markov, para el experto en inteligencia artificial (IA) es preciso tener un mínimo entendimiento de la parte realizada por el lingüista y, por tanto, conocer la esencia de estos campos:

- **Gramática**: que incluye la morfología, la sintaxis y, para algunos autores, también la fonología.
- **Sintaxis**: acorde al texto anteriormente citado de Jurafsky, estudia las reglas y principios que gobiernan la combinación de los constituyentes sintácticos. Analiza la formación de los sintagmas y las oraciones gramaticales. Mediante la sintaxis, se conocen las formas en que se combinan las palabras, así como las relaciones sintagmáticas y paradigmáticas existentes entre ellas. Como los sintagmas pueden unirse, para formar un grupo sintagmático, la sintaxis también establece la manera en la que llevar a cabo el proceso de unificación. En el etiquetado sintáctico se adjudica un **POS** (**Part of Speech**); por ejemplo, en español a la palabra *mostrar* se le asignaría la etiqueta **POS** de *verbo*, mientras que en la oración «el niño llora» se etiquetaría lo siguiente:
    - El ➡️ Determinante.
    - Niño ➡️ Nombre.
    - El niño ➡️ Grupo sintagmático nominal.
    - Llora ➡️ Verbo (y grupo sintagmático predicativo).

    Por supuesto, dentro de una etiqueta pueden existir subcategorías, por ejemplo, «niño» es un sustantivo (nombre) de tipo «común» y «el» es un determinante de tipo «artículo».

Ampliación Existen herramientas web, como la que figura en el siguiente link, que llevan a cabo el etiquetado y la unificación: https://sintaxis.org/analizador/

- **Morfología**: según Jurafsky, estudia las reglas que rigen la flexión, la composición y la derivación de las palabras. Durante el etiquetado morfológico se determina la forma, clase o categoría gramatical de una palabra. El género de las palabras (masculino y femenino), el número de los nombres (singular o plural), o la persona de las formas conjugadas (primera, segunda o tercera) son ejemplos de este campo.

Ampliación El siguiente enlace presenta un analizador morfológico de la Biblioteca Virtual Miguel de Cervantes (BVMC) que usa el corpus Ancora .

https://data.cervantesvirtual.com/analizador

- **Semántica**: siguiendo el texto, estudia significado de las expresiones lingüísticas, es decir, las realidades que representan las grafías.
- **Pragmática**: se centra en el análisis de la relación del lenguaje con los usuarios y las circunstancias de la comunicación o contexto. El contexto debe entenderse como situación, ya que puede incluir cualquier aspecto extra-lingüístico: la situación comunicativa, un conocimiento popular compartido por los hablantes, relaciones personales, y otros muchos.

Importante RESUMEN:

Morfología: El estudio de la información contenida en una palabra considerada ésta en el contexto en el que se utiliza.

Sintaxis: Estudio de las relaciones estructurales entre las palabras en una frase. Estudio de cómo ordenar y agrupar las palabras en la frase.

Semántica: Estudio de los significados de las palabras y su forma de combinarse para formar significados más complejos.

Pragmática, **Discurso**: Estudia cómo el contexto afecta a la interpretación de las oraciones.

El conocimiento inmersivo y exhaustivo de estos términos sus relaciones, y el etiquetado es labor del lingüista, y se lo debe proporcionar al experto en IA para una correcta modelización de un sistema de procesamiento del lenguaje natural, en un idioma concreto.

1.1. Aplicaciones del PLN

Algunas aplicaciones del PLN son:

- Reconocimiento automático del habla
- Traducción automática
- Sistemas de diálogo
- Extracción/recuperación de información
- Interfaces en lenguaje natural
- Herramientas para personas con diversidad funcional
- Ayuda a la redacción
- ...

2. Potencial de las técnicas existentes de procesamiento de lenguaje. Limitaciones

2.1. Potencial del procesamiento del lenguaje natural

Las aplicaciones del procesamiento del lenguaje natural son muchas: chatbots para el desarrollo de los asistentes de compra o sistemas conversacionales, análisis del sentimiento y de la opinión que permite detectar por las palabras empleadas sesgos y opiniones sobre un tema, especialmente en publicaciones de redes sociales, o en encuestas determinadas; clasificación automática de documentos, búsqueda de similitudes en textos para la búsqueda de plagios, detección de entidades (NER) que permiten contextualizar y clasificar textos, búsqueda automáti ca de información en múlti ples idiomas, traducción automática, reconocimiento del habla, y otras muchas.

Los aspectos fundamentales de las aplicaciones arriba mencionadas son los que se explican a continuación.

2.1.1. Reconocimiento del habla (ASR, Automatic Speech Recognition)

Disciplina encargada de convertir los fonemas emitidos por un ser humano en espectros de ondas de audio captados mediante un dispositivo de entrada de sonido de una máquina que, tras ser procesados dentro del contexto de un modelo lingüístico, den lugar a las grafías escritas del mismo; es decir, los sistemas de las máquinas encargados de convertir la voz captada por medio de un sensor (normalmente un micrófono) en la forma escrita de una lengua.

2.1.2. Síntesis de texto a voz (TTS, Text To Speech)

Estos sistemas hacen la labor contraria a un ASR, es decir, a partir de un texto escrito son capaces de reproducirlo mediante el dispositivo de salida de audio (altavoces).

2.1.3. Detección de entidades nombradas (NER, named entity recognition)

Consiste en trozear el texto (tokenizar) de tal manera que se detecten las palabras clave. Se trata pues de una labor de extracción de información que localiza y clasifica en categorías (normalmente personas, organizaciones, lugares, y cantidades) las entidades encontradas en un texto.

2.1.4. Traducción automática

Se centra en el desarrollo de sistemas capaces de traducir automáticamente texto o habla de un idioma a otro.

2.1.5. Similitud de textos

Algunos modelos (normalmente de tamaño considerable) han sido entrenados de tal forma que las palabras son etiquetadas de manera que aludan a conceptos semánticos.

Por ejemplo, un perro es un mamífero y a su vez es un animal. Un etiquetado correcto situará en lugares más próximos entre sí a un perro y a un gato (dos mamíferos) que a un perro y a un salmón. De la misma manera una manzana es una fruta, que a su vez es comida. En buena lógica es más «parecido» semánticamente a una manzana una naranja que una zanca de pollo, aunque todo sea comida.

Mediante técnicas de vectorización y de cálculo del coseno es posible analizar dos textos y decir el «parecido» que tengan entre sí, aunque lógicamente esta apreciación de parecido es subjetiva y depende del set de datos que haya sido empleado para llevar a cabo el entrenamiento.

2.1.6. Análisis del sentimiento

El procesamiento del lenguaje natural puede emplear también el etiquetado característico del aprendizaje automático supervisado para entrenar un modelo, de tal manera que sea capaz de captar la positividad o negatividad de un texto en relación con un tema particular.

Esta potente herramienta es especialmente útil para llevar a cabo sondeos «invisibles» de la opinión de grandes grupos muestrales alrededor de un tema. Las redes sociales, y especialmente Twitter, conforman un campo de datos especialmente bueno para llevar a cabo este tipo de aplicaciones, ya que permiten analizar la opinión de un gran grupo de personas (por ejemplo, todo un país) alrededor de un asunto, o de una persona, lo que indirectamente se puede considerar un excelente sondeo de la opinión sobre la valoración de un político, o de un evento. También puede permitir a un asistente o chatbot, en una conversación larga, averiguar el estado de ánimo del usuario, y actuar en consecuencia.

2.2. Limitaciones. La ambigüedad

Una de las limitaciones mayores alrededor del procesamiento de lenguaje natural es la ambigüedad lingüística. Múltiples interpretaciones de una misma palabra pueden arruinar la capacidad de un sistema automático para responder correctamente y desarrollar su función. El etiquetado y el análisis visto en el apartado anterior ayuda a evitar este tipo de errores. Un ejemplo detallado para el español puede verse en el artículo de C. Zapata et al., de la Universidad de Colombia (Zapata, 2007). Los problemas de ambigüedad más comunes en los textos son los siguientes:

- **Ambigüedad sintáctica**: se presenta cuando una oración tiene asociada más de una representación sintáctica, es decir, si hay más de una regla gramatical representa dicha oración.
- **Ambigüedad léxica/morfológica** (gramatical): la primera ocurre cuando la duda surge respecto a un término aislado, que admite diversas interpretaciones. La segunda tiene lugar si una palabra que se encuentra en una oración representa más de un rol sintáctico o categoría gramatical dentro de la misma.
- **Ambigüedad semántica**: se presenta cuando afecta a un elemento de la frase que puede ser interpretado de diversos modos.
- **Ambigüedad pragmática**: aparece si la realidad depende del contexto del lenguaje y del hablante, en un momento dado.
- **Ambigüedad fonológica**: se presenta cuando una cadena de sonidos puede resultar confusa.
- **Ambigüedad funcional**: se observa si se emplea un término con doble función gramatical.

Ejemplo **Ambigüedades**

- **Ambigüedad sintáctica:**

  Los perros y los gatos enfermos son recogidos por el servicio municipal de recogida de animales. Se podría tener la duda de si son recogidos todos los perros y sólo los gatos enfermos, o si sólo los enfermos (ya sean perros o gatos). Compro los libros baratos. No se puede afirmar si los libros que estoy comprando son los baratos (adjetivo de libros) únicamente, o si estoy comprando libros, que resultan ser baratos (complemento predicativo): http://gedlc.ulpgc.es/investigacion/desambigua/morfosintactico.htm

- **Ambigüedad léxica/morfológica**:
  - Usted aquí no pinta nada (no se sabe si se refiere a que no tiene mando o a que no pinta por ejemplo una pared).
  - Pedro y yo escribimos un cuento (no se sabe si lo hemos escrito ya o lo estamos escribiendo porque la forma conjugada puede pertenecer a un presente de indicativo o a un pretérito).
- **Ambigüedad semántica:**

  Pedro quiere pelearse con un italiano (no se distingue si se trata de cualquier italiano o de un individuo concreto).

- **Ambigüedad pragmática**:

  Golpeó el armario con el bastón y lo rompió (no se sabe si se rompió el bastón o el armario).

- **Ambigüedad fonológica**:

  «es»-«conde» (puede significar una forma conjugada del verbo esconder o el predicado del verbo ser, o un título nobiliario).

- **Ambigüedad funcional:**

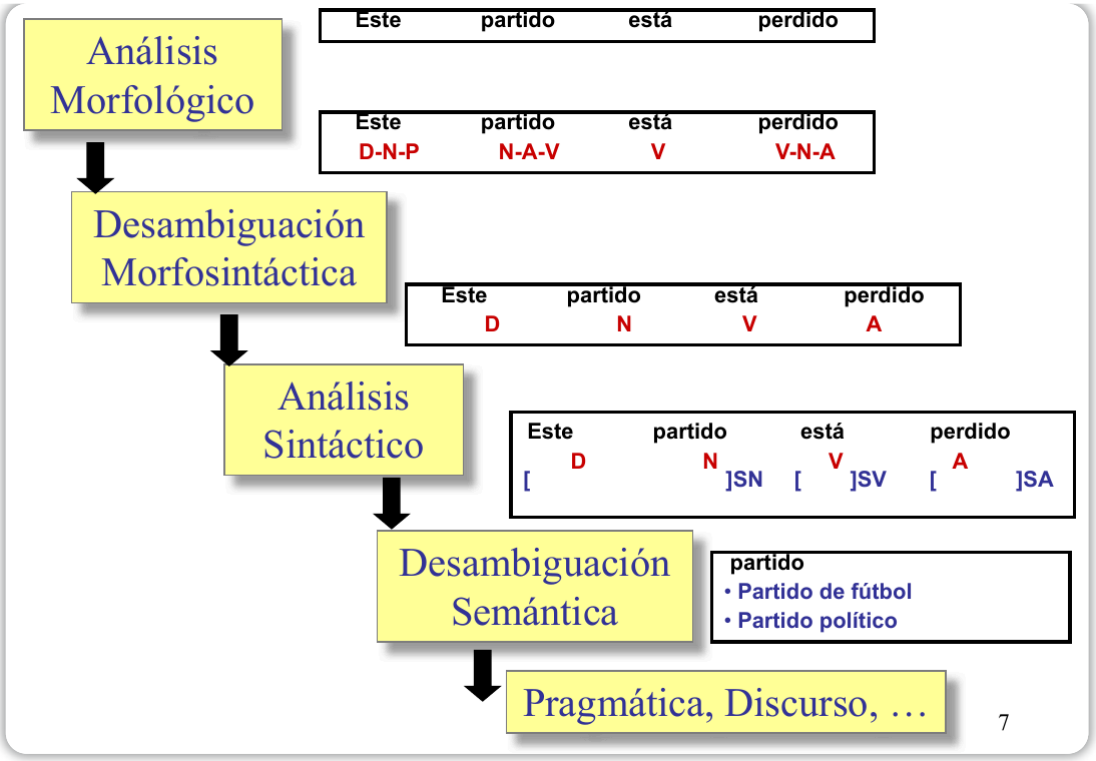
  He vuelto a oler (antes no tenía olfato y ahora sí; o bien, regresé a un lugar a oler algo para comprobar su aroma).

2.3. Desambiguación

La ambigüedad inherente al LN es uno de los problemas presentes en todas fases del procesamiento de la lengua.

Ejemplos:

- Vino de la Rioja.
- Compré unos zapatos de piel de señora.
- La policía observó al sospechoso con unos prismáticos.
- El pescado está listo para comer.
- El cura recibió una cura en su habitación.
- Juan vio a Pedro enfurecido.
- Antonio no nada nada.
- No puedo ir a la fiesta porque no traje traje.
- Estaré de vacaciones solo unos días.
- El Villareal le ganó al Valencia en su campo.
- Me quedé esperándote en el banco.



2.4. POS Tagging

El POS Tagging (Etiquetado morfológico, etiquetado léxico, desambiguación morfosintáctica, ...) es el proceso de asignar, a cada una de las palabras de un texto, la categoría gramatical (POS: part-of-speech), sin tener en cuenta sus relaciones sintácticas. Las palabras, tomadas en forma aislada, en general, son ambiguas respecto a su categoría. La categoría de la mayoría de las palabras se puede desambiguar dentro de un contexto.

Ejemplo

- "Mételo en ese **sobre**" ➡️ nombre
- "Déjalo **sobre** la mesa" ➡️ preposición
- "Dame lo que te **sobre**" ➡️ verbo

2.5. Categorías Gramaticales (POS)

La elección del conjunto de etiquetas afecta a la dificultad del problema. Hay que llegar a un equilibrio entre:

- Obtener la mayor información posible (etiquetas más específicas)
- Simplificar el trabajo de desambiguación (etiquetas más generales)

Tipos de categorías

- Abiertas o Cerradas

Principales Categorías

- Nombres (comunes y propios)
- Pronombres (posesivos, interrogativos, ...)
- Determinantes (artículos, demostrativos, ...)
- Adjetivos
- Verbos
- Adverbios
- Preposiciones
- Conjunciones ...

Ejemplos de Etiquetas (POS)

- Penn Treebank: 45
- Brown Corpus: 87
- Susanne: 350
- LexEsp (PAROLE): 250

Conjunto de etiquetas **Penn Tree Bank**

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	*+,%, &*
CD	Cardinal number	*one, two, three*	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, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	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*			

Conjunto reducido de etiquetas **Parole**

AQ Adjetivos
C0 Conjunción sin clasificar
CC Conjunción Coordinada
CS Conjunción Subordinada
D0 Determinante sin clasificar
DD Determinante Demostrativo
DE Determinante Exclamativo
DI Determinantes Indefinidos
DP Determinante Posesivo
DT Determinante Interrogativo
E0 Términos Extranjeros
I Interjecciones
MC Numeral Cardinal
MO Numeral Ordinal
NC Nombre Común
NP Nombre Propio
P0 Pronombre sin clasificar
PD Pronombre Demostrativo
PI Pronombre Indefinido
PP Pronombre personal
PR Pronombre Relativo
PT Pronombre Interrogativo
PX Pronombre Posesivo
RG Adverbios y Fases Adverbiales
SP Preposiciones
TD Artículos
TI Determinante Indefinido

VAC
V Verbo **A** Auxiliar **C** Condicional
VAG G Gerundio
VAI I Otros tiempos de indicativo
VAM M Imperativo
VAN N Infinitivo
VAP P Participio
VAS S Subjuntivo
VMC M Principal
VMG
VMI
VMM
VMN
VMP
VMS
W Fecha
X Residuales
Y Abreviaturas
Z Cifras
SIGNOS DE PUNTUACIÓN
Faa ¡Fah [Fai¿
Fal { Fap (Fc ,
Fca ! Fcd " Fch]
Fci ? Fcl } Fcp)
Fcs ' Fdp : Fg -
Fgd – Fp . Fpc ; Fps ...
Fs / Ftp % Fac ≪ Fcc ≫

3. Embeddings

Los **embeddings** son representaciones matemáticas (vectores) de elementos como palabras, frases o incluso conceptos, que permiten capturar sus significados o relaciones semánticas en un espacio continuo de menor dimensión. Se utilizan ampliamente en el procesamiento del lenguaje natural (PLN) y en problemas relacionados con datos categóricos.

3.0.1. Características principales de los embeddings:

1. **Dimensiones reducidas**: Los embeddings convierten elementos de entrada (como palabras) de un espacio discreto muy grande (como el vocabulario) a un espacio vectorial denso y de menor dimensión.
2. **Preservación de relaciones semánticas**: Capturan similitudes y relaciones entre palabras basadas en su contexto en los datos. Palabras con significados similares tendrán vectores cercanos en el espacio vectorial.
3. **Adaptabilidad**: Pueden ajustarse durante el entrenamiento para tareas específicas (como clasificación de texto o traducción automática).

3.0.2. Tipos de embeddings más comunes:

1. **Embeddings de palabras**:
    - **Word2Vec**: Genera vectores basados en la idea de que "las palabras que aparecen en contextos similares tienen significados similares". Tiene dos variantes: CBOW (Continuous Bag of Words) y Skip-Gram.
    - **GloVe (Global Vectors for Word Representation)**: Usa estadísticas globales de co-ocurrencia de palabras en un corpus para generar vectores.
    - **FastText**: Extiende Word2Vec al considerar subpalabras, lo que mejora el manejo de palabras desconocidas y morfológicamente ricas.
2. **Embeddings contextualizados**:
    - **ELMo**: Representa palabras dependiendo del contexto en el que aparecen. La palabra "banco", por ejemplo, tendrá diferentes vectores si aparece en un texto sobre finanzas o naturaleza.
    - **Transformers (BERT, GPT, etc.)**: Embeddings generados por modelos que procesan texto completo, entendiendo el contexto a nivel de oración o párrafo.
3. **Embeddings de elementos no textuales**:
    - Embeddings para **categorías**: Representaciones para datos categóricos como productos, usuarios o ubicaciones.
    - **Embeddings gráficos**: Representaciones para nodos o conexiones en redes o grafos.

Ampliación En https://projector.tensorflow.org/ hay una herramienta para la visualización gráfica de word2vec

3.0.3. ¿Cómo funcionan?

Los embeddings son generados mediante redes neuronales o métodos basados en factores matriciales. A través del entrenamiento, se ajustan los vectores para que:

- Palabras o conceptos similares estén más cerca en el espacio vectorial.
- Las relaciones semánticas se capturen. Por ejemplo:
  $\text{vector}(\text{"Rey"}) - \text{vector}(\text{"Hombre"}) + \text{vector}(\text{"Mujer"}) \approx \text{vector}(\text{"Reina"})$

3.0.4. Aplicaciones principales:

1. **Análisis de texto**: Clasificación, búsqueda semántica, análisis de sentimientos.
2. **Traducción automática**: Ayudan a traducir palabras entre idiomas al capturar relaciones universales.
3. **Recomendación**: Embeddings de productos o usuarios para sistemas de recomendación.
4. **Reconocimiento de imágenes**: En visión por computadora, los embeddings se usan para representar características visuales.

Los embeddings han transformado el campo del PLN y otras áreas, ya que permiten que los sistemas comprendan el significado y las relaciones semánticas de las palabras o conceptos de manera eficiente.

4. Derivación y lematización

La **derivación** (stemming) y la **lematización**, que reducen las palabras a sus formas base, pueden ayudar a reducir el número de palabras únicas en los datos, facilitando que el algoritmo identifique relaciones entre ellas.

Tomemos la siguiente oración como ejemplo:
The boys ran, jumped, and swam quickly.
(Los niños corrieron, saltaron y nadaron rápidamente.)

Después de aplicar la derivación, que reduce cada palabra a su forma raíz o base, sin considerar el tiempo verbal o los afijos derivacionales, podríamos obtener:
The boy ran, jump, and swam quick.
(El niño corrió, saltó, y nadó rápido.)

La derivación simplifica el texto a sus formas base. En este ejemplo, "ran," "jumped," y "swam" se reducen a "ran," "jump," y "swam," respectivamente. Observa que "ran" y "swam" no cambian, ya que la derivación a menudo resulta en palabras que están cerca de su forma raíz pero no exactamente en la forma base del diccionario. Este proceso ayuda a reducir la complejidad de los datos textuales, facilitando que los algoritmos de aprendizaje automático coincidan y analicen patrones sin quedar atrapados en variaciones de la misma palabra.

> Importante El idioma utilizado también es clave para la aplicación de la derivación y lematización

Tomemos la siguiente oración como ejemplo:
The boys ran, jumped, and swam quickly.
(Los niños corrieron, saltaron y nadaron rápidamente.)

Después de aplicar la lematización, que considera el análisis morfológico de las palabras, con el objetivo de devolver la forma base o de diccionario de una palabra, conocida como lema, obtenemos:
The boy run, jump, and swim quickly.
(El niño correr, saltar y nadar rápidamente.)

La lematización convierte con precisión "ran," "jumped," y "swam" a "run," "jump," y "swim." Este proceso tiene en cuenta la parte del discurso de cada palabra, asegurando que la reducción a la forma base sea tanto gramatical como contextualmente apropiada. A diferencia de la derivación, la lematización proporciona una reducción más precisa a la forma base, asegurando que el texto procesado permanezca significativo y contextualmente preciso. Esto mejora el rendimiento de los modelos de PLN al permitirles comprender y procesar el lenguaje más efectivamente, reduciendo la complejidad del conjunto de datos mientras mantiene la integridad del texto original.

Otros dos aspectos importantes del preprocesamiento son la normalización de datos y la limpieza de datos. La normalización de datos incluye convertir todo el texto a minúsculas, eliminar la puntuación y estandarizar el formato de los datos. Esto ayuda a asegurar que el algoritmo no trate diferentes variaciones de la misma palabra como entidades separadas, lo que puede llevar a resultados inexactos.

La limpieza de datos incluye eliminar datos duplicados o irrelevantes y corregir errores o inconsistencias en los datos. Esto es particularmente importante en conjuntos de datos grandes, donde la limpieza manual consume mucho tiempo y es propensa a errores. Las herramientas automatizadas de preprocesamiento pueden ayudar a identificar y eliminar errores rápidamente, haciendo que los datos sean más confiables para el análisis.

5. Elaboración de un sistema de procesamiento de lenguaje orientado a una tarea específica

Se propone aunar todo lo aprendido en la generación de un código que tras adquirir vía ASR la conversación de un usuario con un robot de cocina, sea capaz de hacer que este funcione, teniendo en cuenta las siguientes limitaciones:

- El robot puede cocinar mediante calor (cocer). Esta operación requiere de una temperatura en grados Celsius y de un tiempo expresado en minutos, hasta un máximo de 120 °C.
- El robot puede preparar alimentos agitando un instrumento, como por ejemplo batir o remover algo (batir). Esta operación requiere de una velocidad expresada en un valor del 0-10 y de un tiempo expresado en minutos, hasta un máximo de 120.
- El robot puede parar (lo cual implica velocidad de giro 0, temperatura 0 °C, tiempo 0).
- El robot ha de poder entender frases alternativas, es decir, en lugar de batir, emplear el verbo remover, o agitar, o uso de verbos generales como «pon el robot a velocidad 5».
- El robot ha de poder entender frases compuestas, que especifiquen en un solo comando temperatura, velocidad y tiempo.

El abordaje de esta labor es sencillo:

1. Un ASR convierte la voz a texto.
2. El texto es procesado:
    - Se quitan las palabras inútiles (stop words).
    - Se unifican mayúsculas, minúsculas y signos.
    - Se tokeniza y se etiqueta.
    - Se localizan los tokens correspondientes a verbos de acción (cocer, batir).
    - Se localizan los tokens correspondientes a valores numéricos.
    - Se va haciendo agregación sintagmática.
    - Se activan las salidas en una GPIO (véase Unidad 4).

> ##
>
> NLTK y Spacy son librerías simples pero muy eficaces para trabajar con los aspectos informáticos principales del lenguaje natural, si bien NeMo es una solución mucho más potente capaz de llevar a cabo labores TTS y ASR.

El reconocimiento de entidades nombradas (NER), el tokenizado, y el etiquetado son instrumentos básicos a la hora de descomponer un texto y extraer la información más relevante.

En el reconocimiento del lenguaje natural, además de la figura del técnico, ha de existir una figura asociada al lingüista.

6. Formación del investigador en PLN

Un experto en sistemas de inteligencia artificial que desee adquirir conocimientos en procesamiento del lenguaje natural puede seguir muchos itinerarios. El aquí propuesto es una ruta tradicional, basada en una primera formación teórica y clásica, y una posterior aplicación de los métodos de procesamiento del lenguaje natural basados en redes neuronales. Por ello se aconseja la lectura del texto de Jurafsky, y posteriormente la lectura del manual base del Natural Learning ToolKit (NLTK) en lengua inglesa. A partir de ahí, continuar con Spacy en lenguas inglesa y española, y finalmente dar el paso a nVidia NeMo para ejecución de ASR, TTS y NLP en plataformas jetbot y posteriormente crecer a un entorno de servidores tipo JARVIS/TRITÓN.

- **NLTK**: puede ser descargado desde el portal web propietario https://www.nltk.org/ y a partir de ahí leer las instrucciones de instalación https://www.nltk.org/install.html, que son muy simples, y descargar los sets de datos mediante el comando `nltk.download(all)` dentro del entorno de Python 3 (y tras ejecutar el import correspondiente). Incluso puede instalar el Stanford CoreNLP API para NLTK desde https://github.com/nltk/nltk/wiki/Stanford-CoreNLP-API-in-NLTK que incluye modelo para nuestra lengua. Dentro de NLTK debería seguir el orden lógico que se muestra en el NLTK-BOOK público (disponible en https://www.nltk.org/book/):
    - Cargar un corpus.
    - Usar expresiones regulares.
    - Tokenizar y etiquetar.
    - Emplear lo anterior para hacer etiquetado POS.
    - Usar diccionarios de eliminación de stop-words.
    - Usar N-gramas.
    - Lematizar.
    - Programar un clasificador (por ejemplo, de películas según temática o de noticias).
    - Programar un comparador de textos.
    - Programar un analizador de sentimientos.
    - Programar un sistema de ideas clave o resumen.
    - Emplear un TTS (no desde NLTK).
    - Emplear un ASR (no desde NLTK).
    - Programar un procesador semántico real.
- **Spacy**: puede ser descargado e instalado siguiendo las instrucciones de https://spacy.io/ . En él se debería practicar lo anterior, además de:

    - Vectorización y comparativa por vectorización a partir de modelos pre entrenados disponibles en el programa como los empleados en este texto.
    - Lo mismo con otros modelos entrenados, como variaciones de BERT.
    - Generar un modelo propio de vectorizado por etiquetación, con un set muy cerrado, de tal manera que las similitudes sean más específicas al texto que se esté tratando.
    - NER, y análisis morfológico con visualización de grafos a través de displacy.
    - Traducción entre idiomas.
- **NeMo**: ha de ser empleado de manera global para NLP, TTS y ASR siguiendo los tutoriales y ejemplos de su GitHub, destacándose entre otros, los siguientes:
    - ASR:
        - Detección de voz (Voice activity detection) a través de micrófono
        - Voz a Texto en PC x86, offline.
        - Voz a Texto en PC x86, online.
        - Voz a texto en Jetbot.
        - Comandos de voz (la base para un asistente).
    - TTS.
    - NLP:
        - Lo mismo que en Spacy.
        - Uso del modelo de Megatron.
        - Programar un Q&A tipo Jeopardy.
- **jetson-voice**: es una biblioteca de inferencia de aprendizaje profundo ASR/NLP/TTS para Jetson Nano, TX1/TX2, Xavier NX y AGX Xavier. Es compatible con Python y JetPack 4.4.1 o posterior. Los modelos DNN se entrenaron con NeMo y se implementaron con TensorRT para optimizar el rendimiento. Todos los cálculos se realizan utilizando la GPU integrada. Actualmente se incluyen las siguientes capacidades:
    - Reconocimiento automático de voz (ASR)
        - Transmisión ASR (QuartzNet)
        - Reconocimiento de comandos/palabras clave (MatchboxNet)
        - Detección de actividad de voz (VAD Marblenet)
    - Procesamiento del lenguaje natural (PNL)
        - Clasificación de intención conjunta/espacio
        - Clasificación de texto (análisis de sentimiento)
        - Clasificación de tokens (reconocimiento de entidad nombrada)
        - Preguntas/Respuestas (QA)
        - Texto a voz (TTS)

7. Fuentes de información

- Wikipedia
- GhatGPT
- Modelos de Inteligencia Artificial (Ed. Marcombo)
- https://iep.utm.edu/artificial-intelligence/
- Materiales MIA curso MEC-20230524
- Introducción al Procesamiento del Lenguaje Natural (PLN) Ferran Pla 2017