

Introducción

Índice

Introducción	3
0. Objetivos de la unidad	3
1. Introducción	3
1.1. Orígenes de la extracción de información	5
1.2. Información estructurada vs. información no estructurada	5
2. Tareas de extracción de información	6
2.1. Reconocimiento de entidades	8
2.2. Clasificación automática	12
2.3. Extracción de información elaborada	13
2.4. Recuperación de información	16
2.5. Respuesta a preguntas	16
2.6. Extracción de resúmenes	17
3. Generación automática de texto	18
3.1. Fundamentos	18
3.2. Modelos GPT	20
4. Métricas de evaluación	21
5. Aprendizaje automático y deep learning	26
5.1. Aprendizaje automático	26
5.2. Redes neuronales y deep learning	29
5.2.1. Redes neuronales convolucionales	35
5.2.2. Capa de atención	36
5.2.3. Redes neuronales recurrentes	37
5.2.4. Arquitectura encoder-decoder	38
5.2.5. Modelo de transformer	40
5.2.6 Ventajas e inconvenientes	41
5.3. Vectores de embeddings	43
Recursos	48
Enlaces de Interés	48

Introducción



El objetivo de esta unidad es presentar una panorámica general de las tareas de procesamiento del lenguaje natural dedicadas a la extracción de información.

0. Objetivos de la unidad

El objetivo de esta unidad es presentar una panorámica general de las tareas de procesamiento del lenguaje natural dedicadas a la extracción de información. La extracción de información (en inglés *information extraction*, IE) tiene como objetivo extraer automáticamente información estructurada de documentos no estructurados, es decir, de texto libre. La extracción puede realizarse con diferentes niveles de abstracción, que se plasman en distintas tareas de complejidad variable, como el reconocimiento de entidades y conceptos relevantes del texto, la extracción de piezas de conocimiento más elaborado (llamadas *insights*), en forma de grafos semánticos de objetos y relaciones entre objetos, la clasificación automática (es decir, la detección de una o varias etiquetas relevantes para describir el texto, incluyendo análisis de sentimientos, emoción, opinión o reputación), y también técnicas generales de recuperación de información, sistemas de respuesta a preguntas (*question answering*) y extracción de resúmenes.

En esta unidad, el **apartado 1** va a detallar los objetivos de la extracción de información estructurada a partir de textos, sus orígenes, aplicaciones y relación con las tecnologías de procesamiento del lenguaje natural.

En el **apartado 2** se presentarán las diferentes tareas existentes relacionadas con la extracción de información, describiendo de forma breve en qué consisten y las técnicas principales para abordarlas. En próximas unidades del curso se analizará con detenimiento cada una de estas tareas.

En el **apartado 3**, por completitud, se describe la tarea de generación automática de texto, que es la contraria a la extracción de información.

En el **apartado 4** se describirán las métricas de evaluación generales que se emplean habitualmente para evaluar este tipo de sistemas, que son más o menos comunes a todas las tareas.

Y, por último, en el **apartado 5** se realizará primero una introducción a los conceptos de aprendizaje automático para luego centrarse en redes neuronales y aprendizaje profundo (*deep learning*), como caso particular de aprendizaje automático. Asimismo, también atenderemos aquí la técnica de *embeddings* para representación semántica de los textos, por su gran aplicación en el procesamiento del lenguaje natural.

1. Introducción

La cantidad de información disponible en Internet en todas las áreas de conocimiento de la humanidad ha aumentado de forma exponencial en los últimos años. Una de las claves del éxito en esta sociedad del conocimiento es la habilidad para presentar los contenidos de forma atractiva, claramente ordenados, con posibilidad de búsquedas e incluyendo cualquier valor añadido como vínculos a contenidos relacionados, información sobre entidades o eventos involucrados, opiniones en blogs o redes sociales, etc. En este contexto, los sistemas de extracción de información permiten almacenar, procesar, filtrar y organizar ese enorme volumen de datos, para convertirlo en información útil y, finalmente, en conocimiento.



La **extracción de información** (en inglés, *information extraction*, IE) es la tarea de extraer automáticamente información estructurada de documentos no estructurados, es decir, de textos en lenguaje humano, mediante el procesamiento del lenguaje natural.

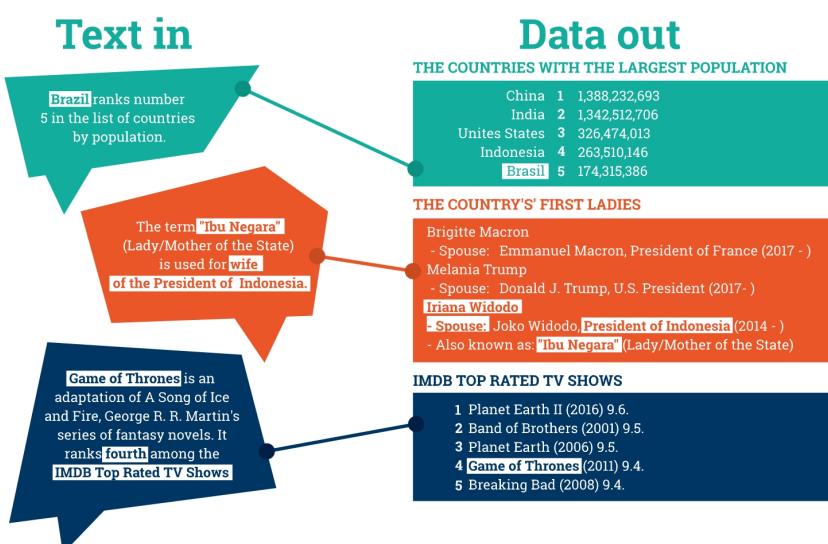


Extracción de información estructurada

Fuente de la imagen: [Ontotext](#)

Generalizando, la extracción de información también se puede aplicar al procesamiento de documentos multimedia para obtener datos estructurados de contenidos de imágenes, audio y vídeo, para lo cual se necesita un preprocesamiento especial que, en general, consiste en convertirlos a texto y luego aplicar las mismas técnicas.

El objetivo de la extracción de información es representar la información semántica de un texto de forma simplificada para poder llevar a cabo tareas de gestión de la información que resulten útiles en un determinado escenario. Esta representación simplificada puede ser de múltiples tipos, según se requiera en cada caso: una lista de entidades y conceptos que aparecen en el texto, las categorías de un modelo de clasificación que mejor definen la temática del texto, o un resumen generado automáticamente de todo el contenido. Por ello, la extracción de información incluye diferentes tareas que generan datos estructurados de diferente naturaleza y nivel de abstracción, que son las que se van a estudiar en este curso.



Tareas de extracción de información

Fuente de la imagen: [Ontotext](#)

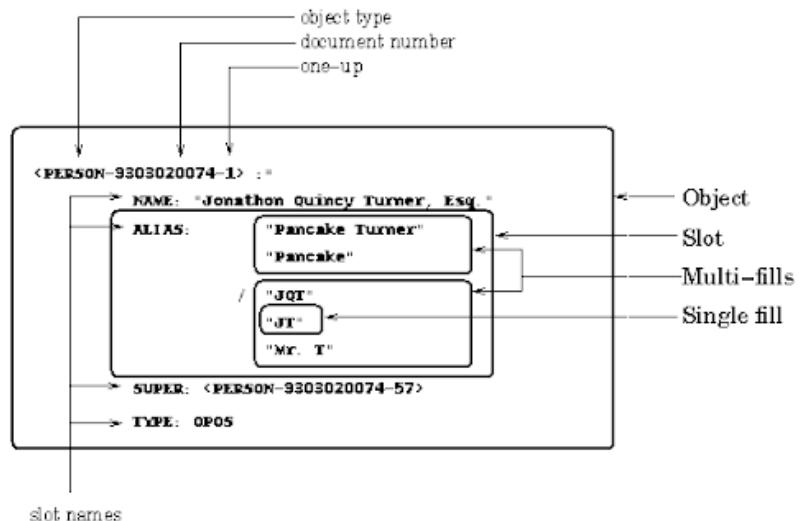


En definitiva, la extracción de información es un área de las tecnologías de procesamiento del lenguaje natural que se ocupa del problema de idear métodos automáticos para la gestión de textos, más allá de su transmisión, almacenamiento y visualización.

1.1. Orígenes de la extracción de información

Históricamente, la extracción de información se remonta a finales de la década de los 70 y principio de los 80, en los primeros tiempos del procesamiento del lenguaje natural. El primer sistema comercial fue JASPER, construido para Reuters por el Carnegie Group Inc. a mediados de los 80 con el objetivo de proporcionar noticias financieras en tiempo real a los operadores financieros [Cowie, Jim; Wilks, Yorick (1996). *Information Extraction*].

A partir de 1987, la agencia DARPA (Defense Advanced Research Projects Agency) de Estados Unidos impulsó una serie de conferencias destinadas a la investigación en comprensión de textos y fomentar el desarrollo de nuevos y mejores métodos de extracción de información, las llamadas MUC o Message Understanding Conference. Hubo 7 ediciones, de 1987 hasta 1998, que supusieron un antes y un después en estas tecnologías.



MUC Scoring Software User's Manual
Fuente de la imagen: SAIC Information Extraction

1.2. Información estructurada vs. información no estructurada

Anteriormente se ha hecho referencia a los términos de información estructurada y no estructurada. A continuación, se van a describir técnicamente estos conceptos.

Información no estructurada

La información no estructurada (o datos no estructurados) es información que no tiene un modelo de datos predefinido o que no está organizada de manera predefinida. Típicamente, este término se utiliza para referirse a texto en lenguaje humano (natural), escrito en un idioma determinado.

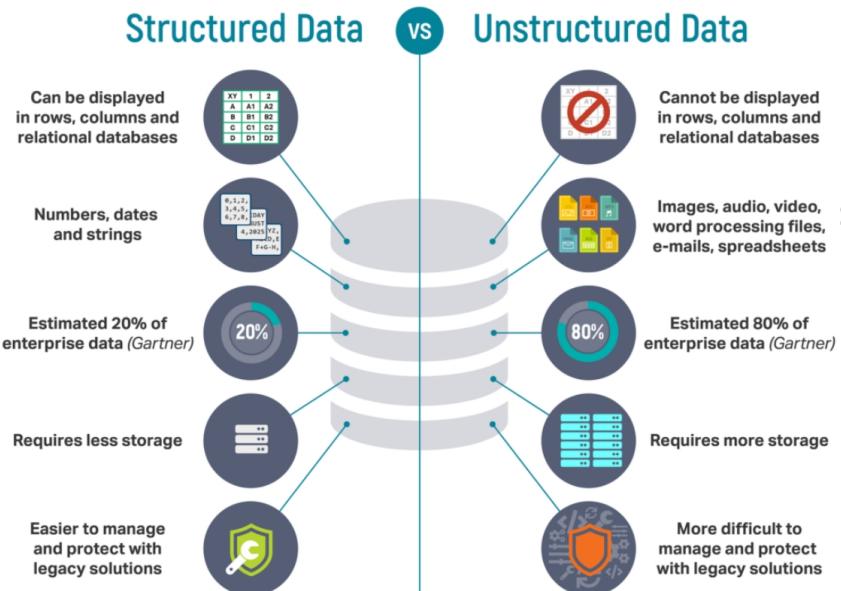
Información estructurada

Por contraposición, la información estructurada son los datos típicos de las bases de datos, caracterizados por tener un esquema determinado que define cómo están organizados, qué nombre y tipo de campos tienen, y cómo se relacionan entre ellos. En definitiva, es información en forma de campos con su nombre, tipo y valor, cuya semántica (significado) está bien definida y que suelen estar normalizados.

“ ”

En 1998, Merrill Lynch afirmó que "los datos no estructurados comprenden la gran mayoría de los datos que se encuentran en una organización, llegando en algunas estimaciones al 80% de todos los datos de la organización" [Christopher C. Shilakes, Julie Tylman (1998). *Enterprise Information Portals*. Merrill Lynch].

Sin embargo, en general se asume que la información estructurada tiene un valor muy superior a la no estructurada porque es la única que permite hacer uso del conocimiento de la organización para diferentes propósitos.



Structured Data vs. Unstructured Data: what are they and why care?

Fuente de la imagen: Lawtomated.com

2. Tareas de extracción de información

Ya se ha mencionado que existen numerosas tareas dedicadas a la extracción de información en diferentes niveles de abstracción y complejidad.

Reconocimiento de entidades

El reconocimiento de entidades tiene como objetivo, como su nombre indica, la identificación y clasificación de las entidades de un texto. El término "entidad" es una simplificación del término "entidad con nombre" (del inglés *named entity*), con lo que esta tarea también se la conoce con el nombre de NER (*Named-Entity Recognition*).

Clasificación automática

La clasificación (o categorización) automática de textos consiste en asignar automáticamente una o varias categorías (o clases) predefinidas de un modelo a un determinado texto, escrito en un lenguaje natural, según su similitud con respecto a otros textos etiquetados previamente que se emplean como conjunto de referencia para entrenar dicho modelo. La lista de categorías representa el contenido semántico del texto según el modelo que se haya utilizado para clasificar.

Extracción de información elaborada

La extracción de información elaborada consiste en convertir el texto en un grafo semántico, que contiene, además de las entidades mencionadas en el texto (igual que antes, con su tipo semántico y relevancia), las relaciones entre ellas (habitualmente incluyendo también un valor de relevancia o confianza).

Recuperación de información

La recuperación de información (en inglés *Information Retrieval*, IR) es el proceso por el que una colección de datos se representa, se almacena y se busca con el fin de descubrir conocimiento como respuesta a una solicitud del usuario (consulta). Un sistema de recuperación de información devolvería la lista de documentos más relevantes respecto a la consulta realizada.

Respuesta a preguntas

La respuesta a preguntas (en inglés *Question Answering*, QA) es una tarea relacionada con la recuperación de información, que en vez de devolver la lista de documentos donde estaría la respuesta a la consulta realizada, devuelve directamente la respuesta solicitada.

Extracción de resúmenes

Un resumen es una versión abreviada o simplificada de un texto dado, que conserva el contenido semántico esencial para entender el texto completo. La tarea de extracción de resúmenes consiste en generar automáticamente el resumen empleando técnicas de procesamiento del lenguaje natural.

En los siguientes apartados se presenta una descripción más detallada de cada una de estas tareas, ilustrando sus objetivos mediante su aplicación al siguiente texto de ejemplo.



Pablo Ruiz Picasso (Málaga, 25 de octubre de 1881 - Mougins, 8 de abril de 1973) fue un pintor y escultor español, creador, junto con Georges Braque, del cubismo. Es considerado desde la génesis del siglo XX como uno de los mayores pintores que participaron en los variados movimientos artísticos que se propagaron por el mundo y ejercieron una gran influencia en otros grandes artistas de su tiempo.

Fuente: [Wikipedia.org](#)

2.1. Reconocimiento de entidades

El reconocimiento de entidades, también denominado extracción o identificación de entidades, tiene como objetivo, como su nombre indica, la identificación y clasificación de las entidades de un texto.

Una entidad es cualquier palabra o secuencia de palabras que se utilice sistemáticamente en el texto para referirse al mismo objeto del mundo real (persona, organización, lugar, producto, etc.), aunque sea con diferentes variantes (nombre completo, solo apellido, pseudónimo, nombre oficial/popular, etc.).

Back in 2000, People Magazine PUBLISHER highlighted Prince Williams' PERSON style who at the time was a little more fashion-conscious, even making fashion statements at times.

Now-a-days the prince mainly wears navy COLOR suits ITEM (sometimes double-breasted DESIGN), light blue COLOR button-ups ITEM with classic LOOK pointed DESIGN collars PART, and burgundy COLOR ties ITEM.

But who knows what the future holds ...

Duchess Kate PERSON did wear an Alexander McQueen BRAND dress ITEM to the wedding OCCASION in the fall of 2017 SEASON.

Ejemplo de reconocimiento de entidades con Spacy

El término "entidad" es una simplificación del término "entidad con nombre" (en inglés, *named entity*), acuñado en la MUC-6 [Grishman, R.; Sundheim, B. (1996). "Message Understanding Conference-6: a brief history". COLING '96]. Por eso esta tarea se suele denominar también "reconocimiento de entidades con nombre" o NER (*Named-Entity Recognition*, en inglés).

Cada entidad detectada se clasifica en una categoría semántica predeterminada (si es una persona, un lugar, una empresa, un producto, una referencia de tiempo...), según la taxonomía empleada por el sistema de NER.

Como salida, el sistema de reconocimiento de entidades genera una lista de las entidades reconocidas en el texto junto con su tipo semántico y, habitualmente, un valor de relevancia o de importancia dentro de ese texto, casi siempre de forma relativa (de 0 a 100%).

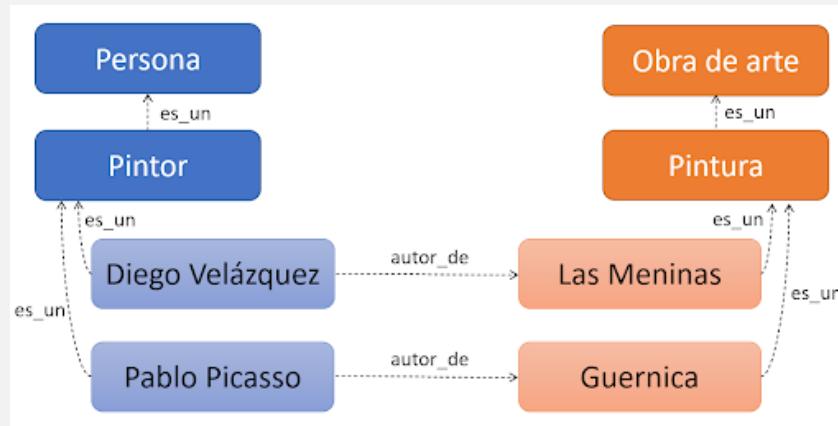
Las entidades no incluyen únicamente nombres propios de objetos como “Pablo Picasso”, “Gran Bretaña” o “Médicos Sin Fronteras” (lo que serían individuos, objetos o *instances* en una ontología), sino que también incluyen nombres comunes que hacen referencia a una clase de objetos, como “pintor”, “país” u “organización no gubernamental” (lo que serían las *clases* en una ontología).

Sin embargo, para aumentar la confusión, algunos sistemas de NER denominan “entidades” (*entities*) únicamente a los individuos y “conceptos” (*concepts*) a las clases, con lo que “Pablo Picasso” sería una entidad y “pintor” sería un concepto.



En la nomenclatura de ontologías, los individuos u objetos (*instances*) son los componentes básicos que se refieren a objetos concretos, es decir, personas, animales, lugares o empresas específicas, llamadas con un nombre único e individual. Por el contrario, una clase (o tipo, categoría, *class*) es una colección de individuos u objetos. Las clases sirven para clasificar individuos u otras clases (subclases).

Por ejemplo, la clase “Persona” podría tener una subclase “Pintor”, que a su vez contiene diferentes individuos entre los que estarían “Pablo Picasso” y “Diego Velázquez”. Y la clase “Obra de arte” podría tener una subclase “Pintura”, que contendrían “Guernica” y “Las Meninas”, relacionadas convenientemente con su autor correspondiente, para formar el grafo semántico completo de la ontología.



Grafo semántico correspondiente a la ontología de ejemplo

Fuente: [Wikipedia.org](#)

Aplicando esta tarea al ejemplo anterior, se obtendría como resultado algo como lo siguiente:



[Pablo Ruiz Picasso|PERSON] ([Málaga|CITY], [25 de octubre de 1881|DATETIME] - [Mougins|CITY], [8 de abril de 1973|DATETIME]) fue un [pintor|VOCATION] y [escultor|VOCATION] [español|LOCATION], [creador|VOCATION], junto con [Georges Braque|PERSON], del [cubismo|ART]. Es considerado desde la génesis del [siglo XX|DATETIME] como uno de los mayores [pintores|VOCATION] que participaron en los variados [movimientos artísticos|ART] que se propagaron por el [mundo|LOCATION] y ejercieron una gran influencia en otros grandes [artistas|VOCATION] de su [tiempo|DATETIME].

O, representado de otra manera, se tendría:

Cálamo Educación S.L.
María Teresa Tijeras Pascual

Cálamo Educación S.L.
María Teresa Tijeras Pascual

Cálamo Educación S.L.
María Teresa Tijeras Pascual

Ejemplo

ENTIDAD	TIPO
Pablo Ruiz Picasso	PERSON
Málaga	CITY
25 de octubre de 1881	DATETIME
Mougins	CITY
8 de abril de 1973	DATETIME
pintor	VOCATION
escultor	VOCATION
español	LOCATION
creador	VOCATION
Georges Braque	PERSON
cubismo	ART
siglo XX	DATETIME
pintores	VOCATION
movimientos artísticos	ART
mundo	LOCATION
artistas	VOCATION
tiempo	DATETIME

Un sistema más avanzado podría incorporar además información de la forma original (lema), indicar si se trata de un individuo concreto (“Pablo Ruiz Picasso”) o bien una clase de individuos (pintores, escultores), o tener una taxonomía de tipos semánticos en varios niveles y referencias a entidades.

Ejemplo

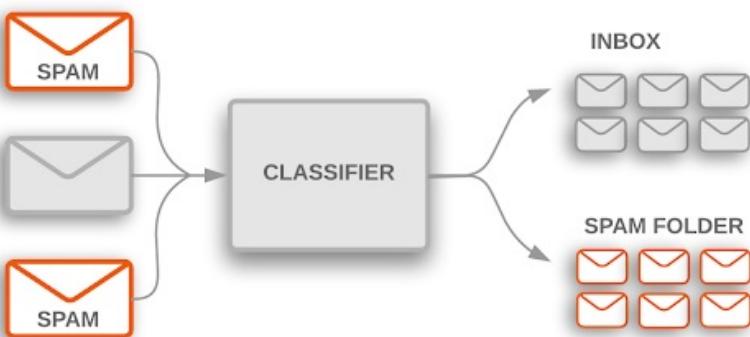
ENTIDAD	TIPO	LEMA
Pablo Ruiz Picasso	PERSON	
Málaga	LOCATION>CITY@España	
25 de octubre de 1881	DATETIME>DATE	
Mougins	LOCATION>CITY@Francia	
8 de abril de 1973	DATETIME>DATE	
pintor, pintores	VOCATION@class	pintor
escultor	VOCATION@class	
español	LOCATION@España	
creador	VOCATION@class	
Georges Braque	PERSON	
cubismo	ART	
siglo XX	DATETIME	
movimientos artísticos	ART	movimiento artístico
mundo	LOCATION	
artistas	VOCATION@class	artista
tiempo	DATETIME	

Como se observa en el ejemplo, únicamente las clases (conceptos) son los que incluyen el lema, puesto que en español solo se flexionan los nombres comunes (en este caso, en género y/o número) y no los nombres propios (que serían entidades).

2.2. Clasificación automática

La clasificación (o categorización) automática de textos consiste en asignar automáticamente una o varias categorías (o clases) predefinidas de un modelo a un determinado texto, escrito en un lenguaje natural, según su similitud con respecto a otros textos etiquetados previamente que se emplean como conjunto de referencia para entrenar dicho modelo. En inglés se denomina *Text Categorization* o *Classification*.

La lista de categorías representa el contenido semántico del texto según el modelo que se haya utilizado para clasificar.

**Detección de SPAM**

Fuente de la imagen: [Text Classification, Google](#)

En este sentido, la utilidad de la lista de categorías sería similar a la de una lista de entidades, aunque extraídas con un proceso un poco diferente. Así, en una aplicación denominada “etiquetado semántico de contenidos”, las dos tareas de reconocimiento de entidades y clasificación automática se combinan para, dado un texto, generar una lista de descriptores o etiquetas descriptivas de su contenido semántico, unas generadas con un proceso de reconocimiento de entidades a partir de las entidades presentes en el texto, y otras generadas con un modelo de clasificación automática.

Podría considerarse que, mientras que el reconocimiento de entidades resalta las menciones de las entidades y conceptos significativos en el texto (de quién o qué habla el texto), la categorización automática especifica sobre qué trata el texto de acuerdo con una taxonomía específica.



Para el ejemplo anterior, un modelo de clasificación temática podría devolver como salida las categorías “Pintura” y “Escultura”, la primera con mayor relevancia porque se menciona dos veces la palabra “pintor” (en singular y plural).

Si se tratara de un modelo de clasificación específica de movimientos artísticos, el sistema devolvería “Cubismo”.

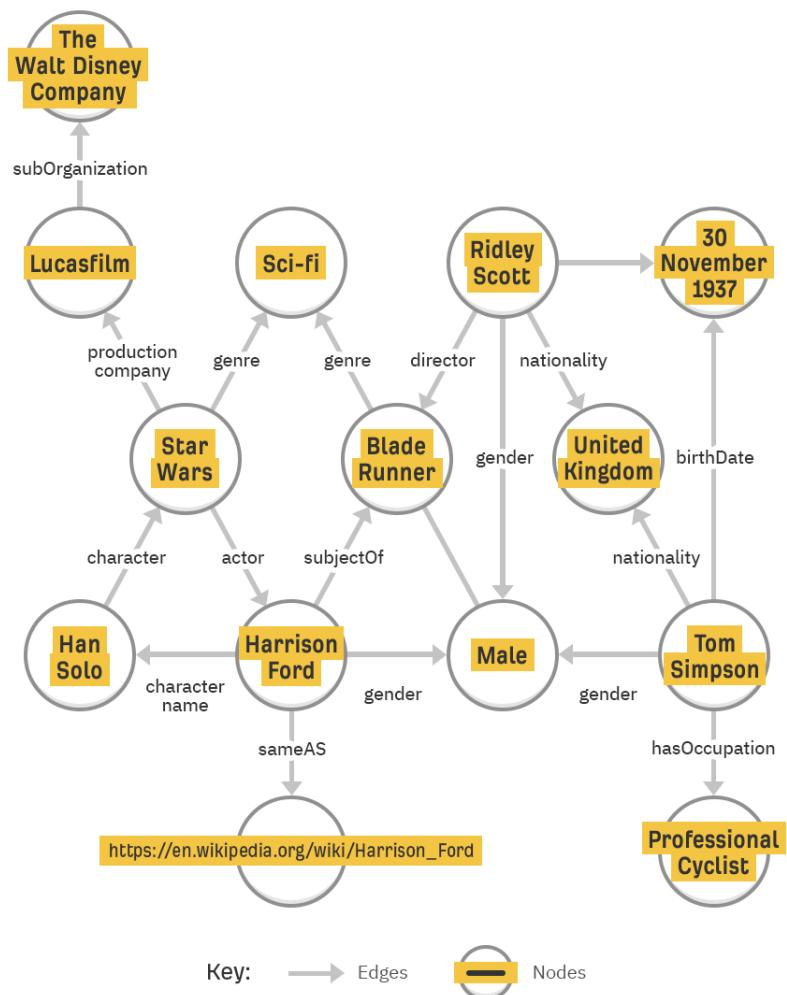
O, mejor, si la taxonomía temática del primer ejemplo incorporara varios niveles, podría categorizarse en “Pintura>Cubismo” siendo más precisos.

Si se tratara de un modelo de clasificación por referencias geográficas, el sistema devolvería “Personajes de España” y “Personajes de Francia” como categorías, la primera con mayor relevancia porque tiene más referencias.

2.3. Extracción de información elaborada

Esta tarea consiste en detectar información estructurada compleja: en vez de limitarse a una lista de entidades, se extraen también las relaciones entre dichas entidades y su significado específico en el texto. En definitiva, el texto se convierte en un grafo semántico, que contiene las entidades mencionadas en el texto (igual que antes, con su tipo semántico y relevancia) y las relaciones entre ellas (habitualmente incluyendo también un valor de relevancia o confianza).

What Google's Knowledge Graph Looks Like



© <https://ahrefs.com/blog/google-knowledge-graph/>

ahrefs

What Google's Knowledge Graph looks like
Fuente de la imagen: [Ahrefs.com](#)

La salida de este sistema es un grafo semántico, que habitualmente se representa con una notación de grafos utilizando alguno de los lenguajes definidos para la Web Semántica como RDF/XML, N-Triples o Turtle.

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
    <dc:title>Tony Benn</dc:title>
    <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

Ejemplo de RDF/XML
Fuente de la imagen: [Wikipedia.org](#)

```

<http://example.org/books/Huckleberry_Finn>
  <http://example.org/relation/author>
  <http://example.org/person/Mark_Twain> .

```

*Ejemplo de Turtle*Fuente de la imagen: [Wikipedia.org](#)

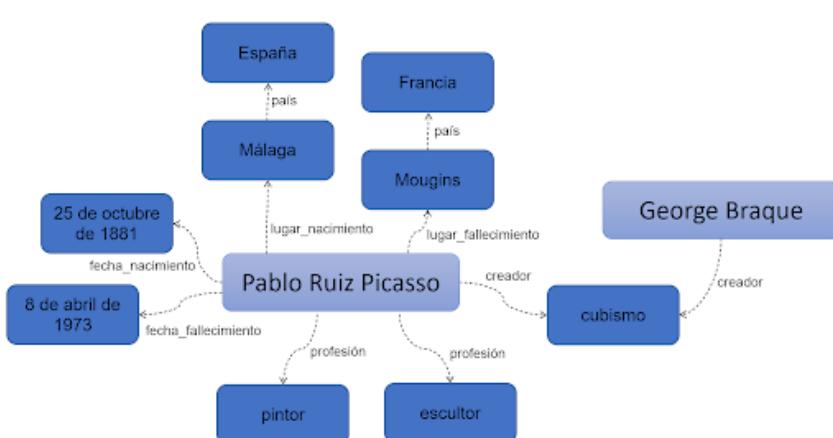
En ocasiones, esta tarea se denomina “extracción de *insights*” (*insight extraction*), utilizando el término *insight*, muy empleado en *marketing* y publicidad, que se refiere a un dato clave, la información esencial y de gran valor que permite comprender o encontrar la solución a un problema.

Para el ejemplo anterior, un sistema de extracción de *insights* podría devolver algo como lo siguiente, añadido a la salida del reconocimiento de entidades.

Ejemplo

Entidad	Atributo	Valor
Pablo Ruiz Picasso	lugar_nacimiento	Málaga
Pablo Ruiz Picasso	fecha_nacimiento	25 de octubre de 1881
Málaga	país	España
Pablo Ruiz Picasso	lugar_fallecimiento	Mougins
Pablo Ruiz Picasso	fecha_fallecimiento	8 de abril de 1973
Mougins	país	Francia
Pablo Ruiz Picasso	profesión	pintor
Pablo Ruiz Picasso	profesión	escultor
Pablo Ruiz Picasso	creador	cubismo
George Braque	creador	cubismo

Esta información puede representarse como el grafo siguiente.

*Grafo semántico del ejemplo anterior*

2.4. Recuperación de información

La recuperación de información (en inglés *Information Retrieval*, IR) es el proceso por el que una colección de datos se representa, se almacena y se busca con el fin de descubrir conocimiento como respuesta a una solicitud del usuario (consulta).



Recuperación de información

Fuente de la imagen: TREC Logo

Un sistema de recuperación de información devolvería la lista de documentos más relevantes respecto a la consulta realizada. Esta consulta habitualmente consiste en una palabra o serie de palabras que además pueden ser escritas según una sintaxis de búsqueda (por ejemplo, con operadores AND, OR y NOT o el + y el - de la sintaxis de Google), pero a veces también puede consistir en uno o varios textos completos, siendo el objetivo de la búsqueda encontrar textos similares a los dados.



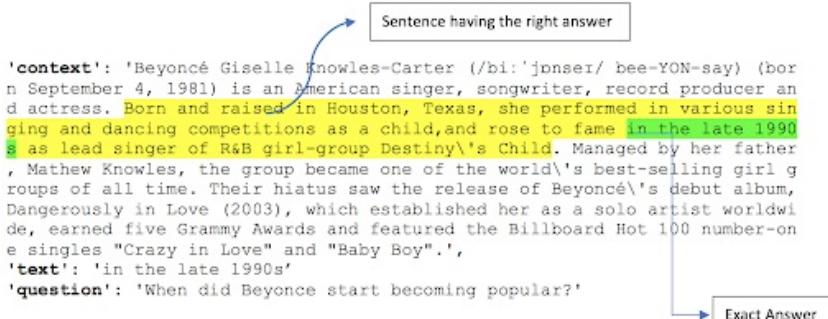
Un sistema de recuperación de información devolvería el texto de ejemplo como resultado de una consulta sobre “cubismo”, o “siglo XX”, ya que son términos que aparecen directamente en el texto.

Si el sistema es más avanzado, y utiliza la información semántica extraída por el reconocimiento de entidades o del grafo semántico, podría encontrarse este texto para la consulta “artistas cuya fecha de nacimiento sea el siglo XIX” o “cubismo en España” (en la sintaxis soportada por el sistema).

2.5. Respuesta a preguntas

La respuesta a preguntas (en inglés *Question Answering*, QA) es una tarea relacionada con la recuperación de información, cuyo objetivo es construir sistemas que respondan automáticamente a las preguntas planteadas en lenguaje natural.

En este caso, en vez de devolver la lista de documentos donde estaría la respuesta a la consulta realizada, el sistema devuelve directamente la respuesta solicitada.



Respuesta a preguntas

Fuente de la imagen: [Towardsdatascience.com](https://towardsdatascience.com/)



Un sistema de *question answering* que utilizaría el texto del ejemplo como base de conocimientos, para la consulta “artistas cuya fecha de nacimiento sea el siglo XIX” devolvería directamente “Pablo Ruiz Picasso”.

O para la consulta “¿dónde falleció Pablo Picasso?” respondería “Mougins, Francia”.

2.6. Extracción de resúmenes

Un resumen es una versión abreviada o simplificada de un texto dado que conserva el contenido semántico esencial para entender el texto completo. La tarea de extracción de resúmenes (*Summary Generation o Extraction*) consiste en generar automáticamente este resumen empleando técnicas de procesamiento del lenguaje natural.

Existen dos tipos de resúmenes: resúmenes extractivos, en los que el sistema selecciona aquellas frases que son más importantes del texto, y resúmenes abstractivos, donde se reescribe (parafrasea) el texto con otras palabras, proporcionando un resumen más rico y elaborado.

Original text:

"...there are two ways to become wealthy: to create wealth or to take wealth away from others. The former adds to society. The latter typically subtracts from it, for in the process of taking it away, wealth gets destroyed. A monopolist who overcharges for his product takes away money from those whom he is overcharging and at the same time destroys value. To get his monopoly price, he has to restrict production."

Stiglitz, J.E. (2013). *The price of inequality*. London: Penguin.

Summary

Stiglitz (2013) suggests that creating wealth adds value to society, but that taking away the wealth of others detracts from it. He uses the example of a monopolist who overcharges for his product resulting in loss of wealth for the customer, but also loss of value as the monopolist has to restrict production in order to charge the higher price.

Extracción de resúmenes

Fuente de la imagen: Libguides.newcastle.edu.au

Aunque el texto de ejemplo es bastante corto, un sistema de extracción de resúmenes extractivo podría seleccionar la primera frase como la más importante, y generar el siguiente resumen:



Pablo Ruiz Picasso (Málaga, 25 de octubre de 1881 - Mougins, 8 de abril de 1973) fue un pintor y escultor español, creador, junto con Georges Braque, del cubismo.

Por contraposición, un sistema de resumen abstractivo podría generar un texto como el siguiente:



Pablo Ruiz Picasso fue un pintor y escultor español, creador del cubismo, que es considerado como uno de los mayores pintores del siglo XX.

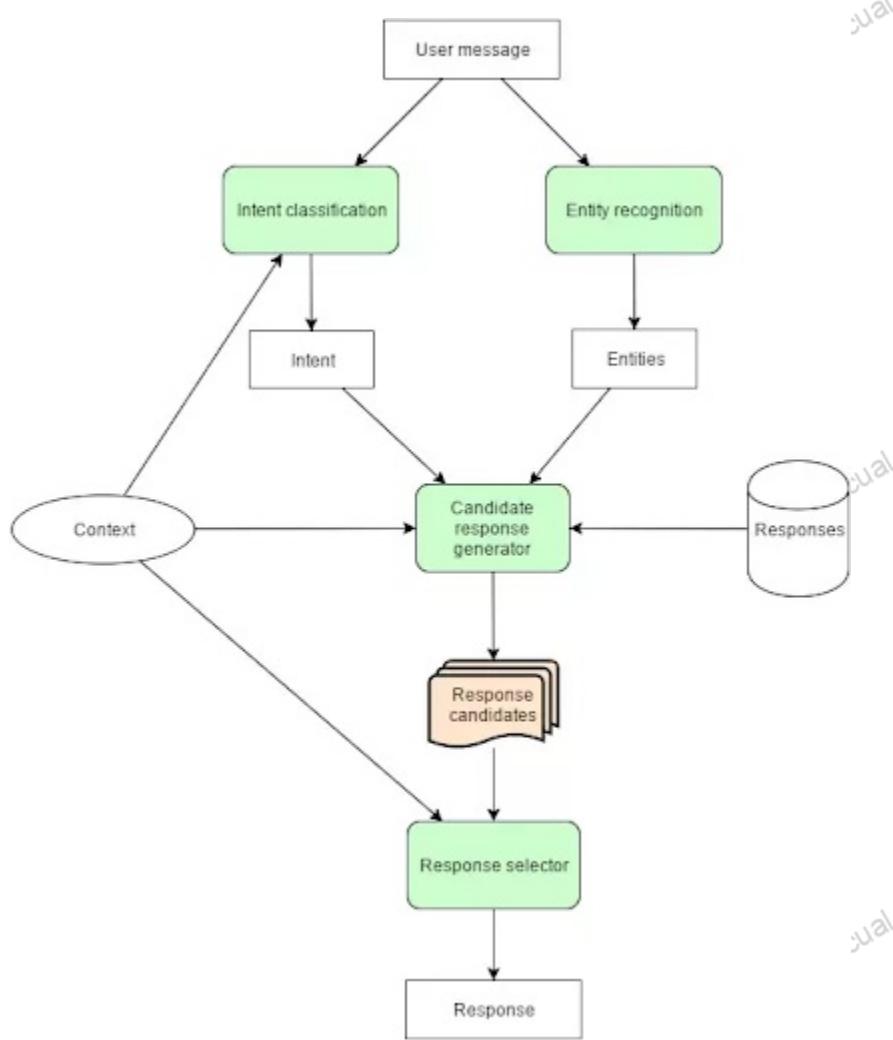
3. Generación automática de texto

3.1. Fundamentos

La generación automática de texto o de lenguaje natural (en inglés, *Natural Language Generation*, NLG) es la tarea opuesta a la extracción de información y consiste en generar automáticamente textos en lenguaje natural que puedan satisfacer determinados requisitos comunicativos, principalmente convirtiendo la información estructurada en información no estructurada, es decir, en texto. En inglés la tarea se denomina *text generation*.

Las aplicaciones más comunes de la generación de texto son la producción de informes, por ejemplo, meteorológicos, la generación de la conversación (mensajes de pregunta/respuesta) en chatbots, la generación de títulos para imágenes (etiquetado automático de imágenes para visión artificial), etc.

Esta tarea existe desde que se desarrolló el sistema ELIZA a mediados de los años sesenta, pero los métodos se utilizaron por primera vez comercialmente en los años noventa. Las técnicas de generación de texto van desde sistemas simples basados en plantillas (como plantillas de mensajes de correo electrónico para generar cartas tipo), hasta sistemas que incorporan una comprensión compleja de la gramática humana, pasando por sistemas basados en aprendizaje automático entrenado sobre grandes corpus, hoy en día utilizando principalmente técnicas de *deep learning*.



Generación de conversación en chatbot
Fuente de la imagen: [Quora.com](#)

Siguiendo con el ejemplo planteado en el apartado anterior, a partir de la información estructurada generada por el sistema de extracción de información elaborada, un sistema de generación de texto podría generar el siguiente texto:



Pablo Ruiz Picasso fue un pintor y escultor que nació en Málaga el 25 de octubre de 1881 y murió en Mougins el 8 de abril de 1973), creador del cubismo con Georges Braque.

3.2. Modelos GPT

OpenAI, un laboratorio de investigación en inteligencia artificial fundado en San Francisco (EE. UU.) por Elon Musk, Sam Altman y otros, publicó en 2018 un documento titulado “Improving Language Understanding by Generative Pre-Training” donde presentaron su modelo GPT (Generative Pre-trained Transformer).

El nuevo modelo propuesto se basaba en *deep learning* con la arquitectura de *transformers* que había sido desarrollada por Google un año antes. El uso de esta arquitectura, a diferencia de las técnicas anteriores de *deep learning* que utilizaban redes neuronales recurrentes con atención (ver apartado 5 más adelante), proporcionó a GPT unas capacidades de aprendizaje y memoria mucho mayores que las que lograban con los mecanismos anteriores, siendo un modelo capaz de llevar a cabo diferentes tareas relacionadas con la generación de texto (traducción, generación de resúmenes, respuesta a preguntas) sin necesidad de un proceso de ajuste a la tarea concreta.

GPT-2, publicado en 2019, es la evolución del primer modelo, una versión 10 veces mayor (1.500 millones de parámetros entrenados con 45 millones de páginas web, en total 40 GB de datos). Su código fuente fue liberado al público, con lo que su uso se hizo muy popular.

El modelo más reciente es GPT-3, publicado en mayo de 2020, mucho más potente, con 4 versiones (Ada, Babbage, Curie, y Davinci) cada una entrenada con diferente número de parámetros, llegando a los 175 mil millones de parámetros (de Davinci). Sin embargo, en esta ocasión su código fuente no fue liberado, sino que el acceso se proporciona a través de una API cerrada (ofrecida por Microsoft) y con un esquema de pago por uso.

En protesta contra los modelos cerrados, en julio de 2020 un grupo de investigadores crearon EleutherAI, un grupo de trabajo que tiene como objetivo crear inteligencia artificial de código abierto, y uno de los primeros problemas que decidió abordar fue la creación de modelos alternativos a GPT-3 que fueran accesibles libremente.

En marzo de 2021 han publicado GPT-Neo, con 3 versiones: 125 millones de parámetros, 1.300 millones (equivalente a GPT-3 Babbage) y 2.700 millones de parámetros. En junio de 2021 han publicado GPT-J, con 6.000 millones de parámetros, lo que lo convierte en el modelo de código abierto más avanzado hasta el momento, y equivalente directo de GPT-3 Curie.

Las pruebas realizadas con estos modelos (en idioma inglés) muestran un gran rendimiento. Al generar un texto con GPT-J, es casi imposible saber si lo ha escrito un humano o una máquina.

Para utilizar un modelo GPT, simplemente hay que proporcionar un *prompt* (comando) como entrada, expresando lo que se desea obtener, que el modelo completa generando texto relacionado.



En las páginas [Write With Transformer](#) y [Talk To Transformers](#) se pueden probar estos modelos.

La figura siguiente muestra el resultado del *prompt* “what is NLP?”, que pregunta por una definición.

The screenshot shows the InferKit DEMO interface. On the left, there's a sidebar with "Generate Options" and "Advanced Settings". Under "Generate Options", there's a text input field containing "what is NLP?", a length slider set to 200, and a checkbox for "Start at beginning". Under "Advanced Settings", there are checkboxes for "Pause at end" and "Sampling temperature", and sliders for "Nucleus sampling top p" (set to 0.9) and "Sampling temperature" (set to 1). A "Reset" button is also present. In the center, a preview window displays the generated text: "Natural language processing (NLP) is the science of computers and algorithms that can interpret written or spoken words. It may sound like something out of a futuristic science fiction novel, but". Below the preview is a "Generate Text" button.

Resultados de generación para “what is NLP?”

🔗 Buscando en Google por algo como “gpt3 prompt” se encuentran muchos resultados con ejemplos de *prompts* que generan resultados interesantes.

- [GPT-2](#).
- [GPT-3](#).
- [Modelos de EleutherAI](#).

4. Métricas de evaluación

Aunque cada tarea utiliza su propio conjunto de métricas para evaluar la calidad de los resultados y comparar diferentes sistemas, en general en procesamiento del lenguaje natural se utilizan unas métricas habituales de referencia, que son la precisión (en inglés, *precision*), la cobertura (en inglés, *recall*) y la medida-F (en inglés, *F-score*).

Estas métricas provienen del campo de la recuperación de información y se definen normalmente en función de la llamada matriz de confusión. La matriz de confusión es una tabla que permite la visualización del rendimiento de un sistema, presentando en filas frente a columnas el número de predicciones que se han realizado de una categoría frente al número de predicciones que se deberían hacer realmente. Su nombre proviene del hecho de que facilita ver si el sistema está confundiendo entre diferentes clases (es decir, etiquetando comúnmente una como otra).

Por ejemplo, supongamos que se construye un sistema para asignar de forma automática un grado de prioridad a los mensajes recibidos por el servicio de atención al cliente, asignando Alta, Normal o Baja prioridad. Este sistema se ejecuta sobre 3.000 mensajes (1.000 de cada tipo), y se construye la siguiente matriz de confusión.

		Valor Predicho		
		Alta	Normal	Baja
Valor Real	Alta	600	300	100
	Normal	10	900	90
	Baja	50	150	800

Cada fila muestra los valores reales del conjunto de test, mientras que en columnas se muestran las predicciones del sistema. Por ejemplo, de los 1.000 casos de prioridad Normal, 900 fueron clasificados por el sistema correctamente como Normal, mientras que 90 de ellos fueron clasificados como prioridad Baja y 10 casos como prioridad Alta.

La diagonal muestra los aciertos del sistema, aquellos casos donde el sistema clasifica correctamente: 600 ejemplos en prioridad Alta, 900 en prioridad Normal y 800 en prioridad Baja.

Los diferentes casos de acierto/error se denominan como sigue:

Verdaderos Positivos

Los verdaderos positivos o TP (en inglés, *True Positives*) son aquellos casos en los que el sistema acierta. Son los casos de la diagonal de la matriz de confusión.

Se pueden calcular por cada clase individualmente o de forma global. En nuestro ejemplo:

- $TP(\text{Alta}) = 600$
- $TP(\text{Normal}) = 900$
- $TP(\text{Baja}) = 800$
- $TP = 600+900+800 = 2.300$

Falsos Positivos

Los falsos positivos o FP (en inglés, *False Positives*) son los casos que el sistema clasifica incorrectamente en una clase. Son la suma de las columnas que no coinciden con la diagonal. En nuestro ejemplo:

- $FP(\text{Alta}) = 10+50 = 60$
- $FP(\text{Normal}) = 300+150 = 450$
- $FP(\text{Baja}) = 100+90 = 190$
- $FP = 60+450+190 = 700$

Falsos Negativos

Los falsos negativos o FN (en inglés, *False Negatives*) son los casos que el sistema debería haber clasificado en una cierta clase, pero no lo ha hecho. Son la suma de las filas cuando no coincide con la diagonal. En nuestro ejemplo:

- $\text{FN(Alta)} = 300+100 = 400$
- $\text{FN(Normal)} = 10+90 = 100$
- $\text{FN(Baja)} = 50+150 = 200$
- $\text{FN} = 400+100+200 = 700$

Verdaderos Negativos

Por completitud, los verdaderos negativos o TN (en inglés, *True Negatives*) son aquellos casos en que el sistema correctamente no clasifica un ejemplo como una clase determinada. Son aciertos del sistema, pero por omisión.

En este ejemplo, como el sistema siempre asigna una clase a cada ejemplo, el número de TN es 0, ya que o acierta (es TP) o falla (FN para la clase del ejemplo y FP para la clase por la que ha fallado).



Para saber más sobre el concepto de *confusion matrix*, puedes pinchar en este [enlace](#).

A partir de la matriz de confusión se pueden definir un gran número de métricas de evaluación de sistemas de procesamiento del lenguaje natural. Las métricas más importantes son las siguientes:

Precisión

La precisión (en inglés, *precision*) mide el acierto del sistema, es decir, en qué porcentaje la etiqueta predicha es correcta respecto al total de casos predichos para esa etiqueta.



$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Toma valor entre 0 y 1, y es mejor cuanto mayor sea este resultado (más próximo a 1).

Se puede calcular por cada clase:

- $\text{precision(Alta)} = 600/(600+60) = 91\%$
- $\text{precision(Normal)} = 900/(900+450) = 67\%$
- $\text{precision(Baja)} = 800/(800+190) = 81\%$

También se puede calcular globalmente, de dos maneras. Si los cálculos de verdaderos/falsos positivos/negativos se realizan directamente, de forma global, independientemente de la etiqueta, se denominan métricas macro-promediadas (*macro-averaged*). Si los cálculos se realizan primero para cada categoría individual y luego se promedian, se denominan métricas micro-promediadas (*micro-averaged*). Las métricas macro dan una idea del rendimiento global del sistema, y las micro se centran en las categorías individuales y son más apropiadas si las clases están muy desbalanceadas.

Así:



- $\text{precision_micro} = 2.300 / (2.300 + 700) = 77\%$
- $\text{precision_macro} = (\text{precision(Alta)} + \text{precision(Normal)} + \text{precision(Baja)}) / 3 = 80\%$

Cobertura

La cobertura (en inglés, *recall*) mide cuántas categorías se clasifican como tal respecto de las que se deberían haber clasificado.



$$\text{recall} = \text{TP} / (\text{TP} + \text{FN})$$

De igual manera, toma valor entre 0 y 1, y es mejor cuanto mayor sea este resultado (más próximo a 1).

Para el ejemplo anterior:



- $\text{recall(Alta)} = 600 / (600 + 400) = 60\%$
- $\text{recall(Normal)} = 900 / (900 + 100) = 90\%$
- $\text{recall(Baja)} = 800 / (800 + 200) = 80\%$
- $\text{recall_micro} = 2.300 / (2.300 + 700) = 77\%$
- $\text{recall_macro} = 77\%$

Medida-F

Existe una relación inversa entre la precisión y la cobertura, y normalmente cuando se aumenta la calidad (precisión), disminuye la cantidad (cobertura), y al revés. Para encontrar un equilibrio entre ambas, existe la medida-F (*F-Score* o *F-measure*) es la media geométrica entre precisión y cobertura, habitualmente ponderadas con igual peso (en este caso se denomina medida F1):



$$F = 2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$$

Se utiliza para indicar un compromiso (balance) del sistema entre ambas métricas. Toma valor entre 0 y 1 y, como es evidente, es mejor cuanto mayor sea este resultado (esto es, más próximo a 1).

Se calcularía para el ejemplo con la fórmula anterior.

Exactitud

La *accuracy*(o exactitud) es otra métrica combinada, que también tiene en cuenta los verdaderos negativos, y mide si el clasificador devuelve exactamente lo que se supone que debe devolver. Se define como la proporción de predicciones correctas (tanto TP como TN) entre el número total de casos.



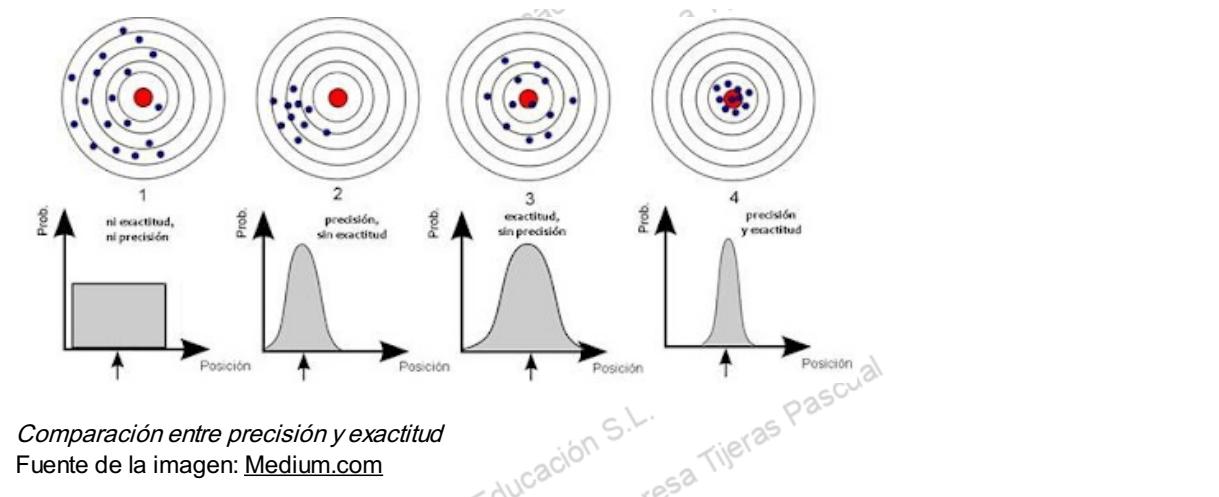
$$\text{accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Para el ejemplo:



- $\text{accuracy_micro} = (2.300 + 0) / (2.300 + 700 + 700) = 81\%$
- (similar cálculo para macro)

No es lo mismo exactitud que precisión, como se muestra en la siguiente figura.



Hay otras muchas métricas, como *sensitivity* (o *true positive rate*) , *specificity* (o *true negative rate*) , *fall-out*, *prevalence threshold*, etc. que se usan en casos específicos.



[Precision and recall.](#)

[La matriz de confusión y sus métricas.](#)

[Performance Metrics for Text Categorization.](#)

Además, existen otras métricas específicas para cada una de las tareas de procesamiento del lenguaje natural, que se describirán cuando sea necesario en las unidades correspondientes.

5. Aprendizaje automático y *deep learning*

5.1. Aprendizaje automático

El objetivo de los sistemas de inteligencia artificial es incorporar la cualidad humana del aprendizaje, esto es, ser capaz de adquirir nuevos conocimientos a partir de las experiencias. Así, el aprendizaje automático, o aprendizaje computacional (del inglés *machine learning*), es la rama de la inteligencia artificial que estudia los algoritmos informáticos/matemáticos que proporcionan la capacidad a los ordenadores de aprender automáticamente a partir de datos de ejemplo. En otras palabras, consiste en generar reglas a partir de ejemplos.

Aunque existen diferentes tipos de aprendizaje, en esta tarea de reconocimiento de entidades se hace referencia al llamado “aprendizaje supervisado”, que consiste en que los algoritmos de aprendizaje construyen un modelo basado en datos de ejemplo, conocidos como “datos de entrenamiento”, con el fin de hacer predicciones sobre nuevos datos.



Los datos de entrenamiento consisten en una matriz de datos organizados en filas y columnas.

- Cada fila es un caso de ejemplo. Cuantos más ejemplos haya, en teoría mejor (más preciso) será el aprendizaje.
- Cada ejemplo está representado por diferentes variables (atributos, características, propiedades, *features*), una en cada columna, que pueden ser numéricas o nominales (etiquetas, por ejemplo “rojo”, True o “clase-A”).

En aprendizaje supervisado, unas variables son las “entradas” y otra variable será la “salida”, de tal forma que el objetivo del sistema es aprender un modelo (función) que calcule (prediga) la variable de salida en función de las entradas.

Cuando la variable de salida es una variable numérica, se denomina “problema de regresión”, mientras que si la variable es de tipo nominal (es decir, una categoría o etiqueta), se denomina “problema de clasificación”

En vez de aprendizaje supervisado, actualmente en el campo del análisis de datos o ciencia de los datos, se utiliza más el término de “técnicas predictivas y de clasificación”.

Por ejemplo, podemos tener el siguiente conjunto de datos de entrenamiento (un fragmento), donde “x1” y “x2” son las variables de entrada e “y” es la variable de salida (o variable objetivo), de tal forma que “y = f(x)”, es decir, “y” se calcula en función de las variables de entrada.

Ejemplo

x_1	x_2	y
1	1	1
1	2	2
3	4	12
5	2	10
6	4	24
7	7	49
8	4	32
9	8	72

Intuitivamente, se puede ver que este conjunto de datos representa la multiplicación, es decir:

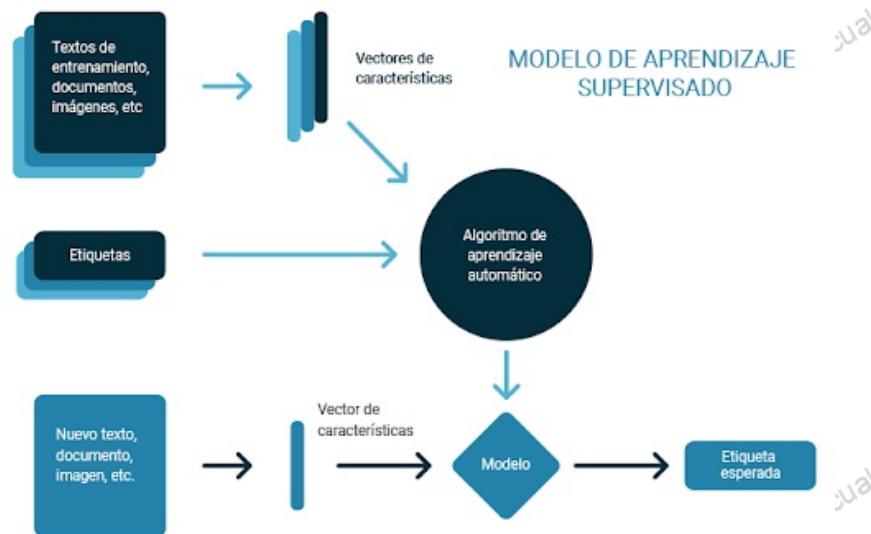


$$y = x_1 * x_2$$

Los diferentes algoritmos de aprendizaje supervisado intentarán obtener esta función, u otra forma de realizar este cálculo.

Existen numerosos algoritmos diferentes, es decir, formas diferentes de abordar el cálculo de esa función, entre los que cabe mencionar los algoritmos de regresión (lineal, polinómica, logística...), vecinos más cercanos (*K-nearest neighbour*, KNN), los árboles de decisión (*decision trees*), las redes bayesianas, las redes neuronales (*neural networks*), máquinas de vectores soporte (*support vector machines*, SVM), etc.

El esquema general de un modelo de aprendizaje supervisado se muestra en la figura siguiente. Los datos de entrenamiento, representados por unos vectores de características, se utilizan para construir un modelo predictivo (la función anterior) mediante uno de los algoritmos de aprendizaje supervisado, que luego se utiliza para calcular el valor de salida (valor número o etiqueta de clasificación) de nuevos ejemplos.

**Aprendizaje supervisado**Fuente de la imagen: [Medium.com](https://medium.com/@mediumsupport/what-is-supervised-learning-101-10e3a2a2a2d)

Como metodología habitual, el conjunto de datos etiquetados (corpus) se divide en dos partes, una de las cuales (en torno al 75-80% de los ejemplos) se utiliza para entrenar el modelo, y la otra (el 20-25% restante) se utiliza para evaluar los resultados del modelo. Esta técnica se denomina *train-test split*.

Una técnica alternativa es la validación cruzada (*cross validation*), mejor desde el punto de vista estadístico ya que permite solucionar el posible error de muestreo al dividir los datos entre *train* y *test*. Funciona dividiendo el conjunto de datos en conjuntos de ejemplos aleatorios de igual longitud (por ejemplo, 4 conjuntos con el 25% de los datos). Para cada conjunto, se entrena un modelo con las muestras restantes (por ejemplo, el 75% de las muestras). A continuación, los modelos realizan las predicciones sobre sus respectivos conjuntos y los resultados se evalúan comparando con las etiquetas anotadas manualmente. El acierto (precisión) global del sistema es el acierto promedio de los diferentes procesos de entrenamiento.

5.2. Redes neuronales y *deep learning*

El aprendizaje profundo o *deep learning* es un caso particular de aprendizaje supervisado que se basa en modelos de redes neuronales, llevados a su máxima expresión de complejidad, que los últimos años ha supuesto una revolución mejorando los resultados de todos los sistemas en todas sus áreas de aplicación.

Para diferenciar entre uno y otro, últimamente se tiende a denominar aprendizaje automático “clásico” o “tradicional” a aquellos algoritmos que no se basan en redes neuronales (o, al menos, no en redes neuronales complejas).

Las redes neuronales artificiales (llamadas simplemente redes neuronales) son modelos inspirados en las redes neuronales biológicas que forman el cerebro. Consisten en un conjunto de nodos (llamados neuronas) interconectados.

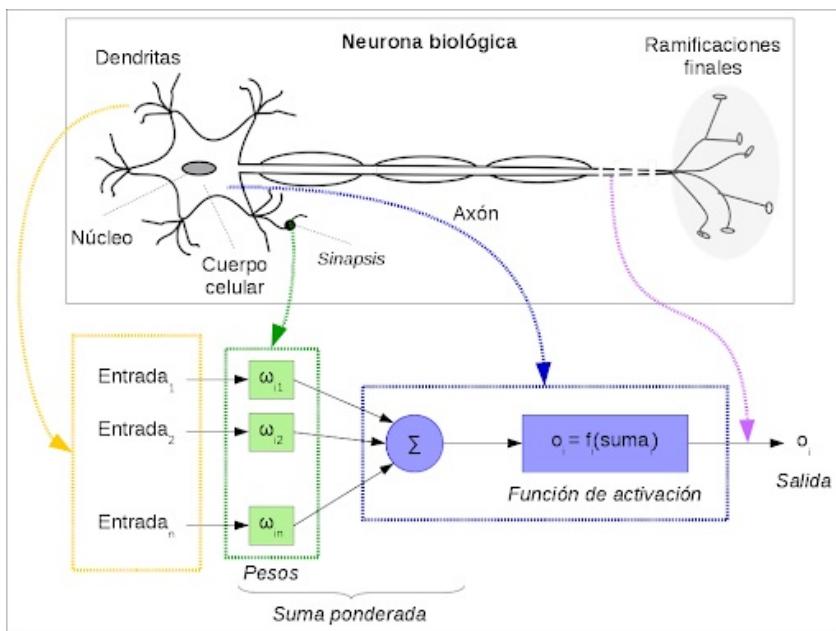
Una neurona es una célula y componente principal del sistema nervioso, cuya función principal es recibir, procesar y transmitir información a través de señales químicas y eléctricas gracias a la excitabilidad eléctrica de su membrana celular (parte externa de la célula). Para llevar a cabo estas funciones, las neuronas tienen un cuerpo celular, donde se realiza el procesamiento de la información; una o varias prolongaciones cortas que generalmente transmiten impulsos eléctricos de entrada hacia el núcleo, denominadas dendritas; y una prolongación larga, denominada axón, que conduce los impulsos eléctricos de salida desde el núcleo hacia otra neurona.

Las neuronas están interconectadas entre sí (en una conexión llamada sinapsis, donde el axón de una neurona se une con las dendritas de las neuronas siguientes), formando una red neuronal. Se supone que la inteligencia y capacidad de raciocinio de los seres vivos (y en particular los seres humanos) está representada (“almacenada” o “codificada”) en las neuronas y las conexiones entre ellas.

Se estima que hay zonas del sistema nervioso, como los órganos de los sentidos y las zonas erógenas, donde cada neurona está conectada hasta con otras 60.000 neuronas, y se estima también que el sistema nervioso de un adulto medio tiene alrededor de 10^{10} neuronas (más que estrellas en la Vía Láctea).



Para saber más sobre la neurona, puedes pinchar en este [enlace](#).



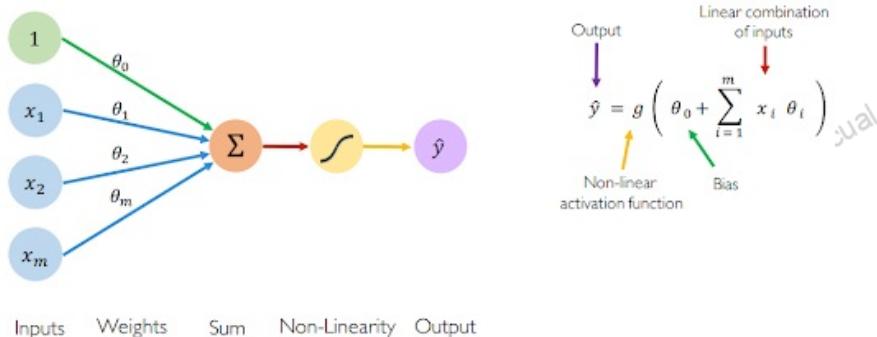
Neurona biológica y modelo matemático

En aprendizaje automático, el algoritmo o modelo de “redes neuronales” está inspirado en el modelo biológico, es una simplificación matemática basada en el funcionamiento estímulo-respuesta de las neuronas biológicas, siendo capaz de “aprender” (modelar) la relación existente entre sus entradas y salidas.

La red neuronal más sencilla se llama “perceptrón” y se compone de una única neurona. El modelo matemático del perceptrón es una función lineal que consiste en un sumatorio de las variables de entrada multiplicada por coeficientes o pesos:



$$y = w_0 + w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n$$



*Modelo matemático del perceptrón
Fuente de la imagen: Kdnuggets.com*

Es decir, cada variable de entrada (que corresponde con la conexión con la salida de otra neurona anterior) tiene un coeficiente asignado. El modelo del perceptrón multiplica las variables de entrada recibidas a través de cada conexión por el coeficiente correspondiente, suma todos los valores, obteniendo así un valor numérico.

Existen numerosos modelos de redes neuronales, dependiendo de características como la topología de conexión de las neuronas, o el tipo de datos de entrada que procesan.

El modelo más habitual es el **perceptrón multicapa**, donde las neuronas están organizadas en capas, de modo que las salidas de las neuronas de una capa actúan como entradas de la capa siguiente. Cada nodo recibe información de los nodos de la capa anterior a los que está conectado y, cuando termina de procesar la información recibida, la envía a la siguiente capa de la red.

El perceptrón multicapa se compone de:

Capa de entrada

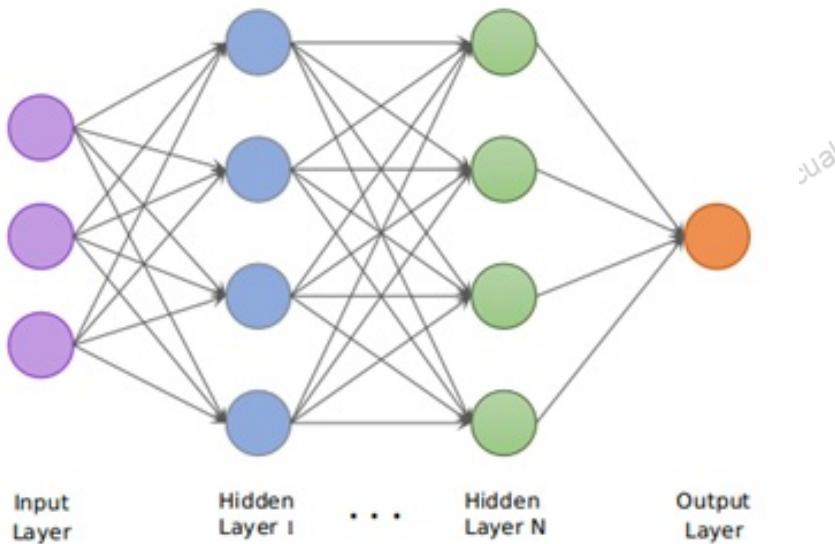
Una capa de entrada (*input layer*), con tantas neuronas como atributos de entrada tenga el conjunto de datos.

Capas ocultas

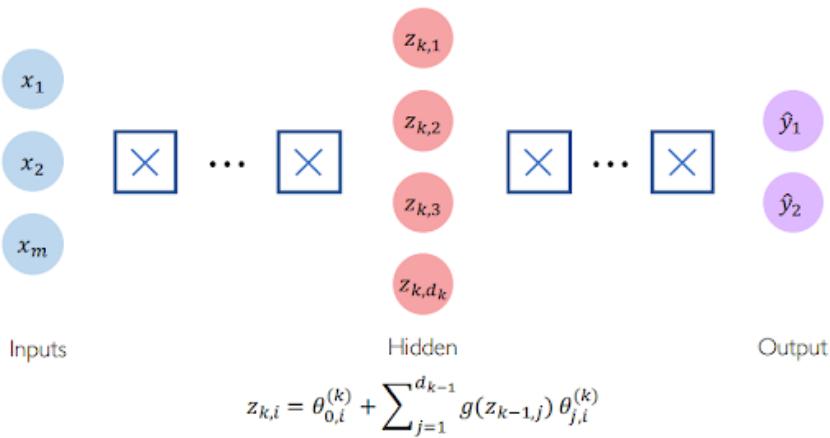
Una o varias capas ocultas (*hidden layers*). El número de neuronas de las capas intermedias es libre. No obstante, hay que tener en cuenta que cuanto mayor sea la cantidad de neuronas ocultas, mayor volumen de datos se necesita para un correcto entrenamiento.

Capa de salida

Una capa de salida (*output layer*), con una única neurona, cuya salida es el valor a predecir para la variable de salida.

*Perceptrón multicapa*Fuente de la imagen: Kdnuggets.com

El aprendizaje profundo o *deep learning* consiste en un variado conjunto de técnicas de aprendizaje automático basadas en redes neuronales con un elevado número de capas (de ahí lo de *deep*), que permiten extraer de forma progresiva características semánticas de alto nivel a partir de los datos de entrada. Se popularizaron hace 3 o 4 años y han supuesto una revolución en todos los campos en los que se ha aplicado.

*Redes neuronales profundas*Fuente de la imagen: Kdnuggets.com

El término “*deep learning*” fue propuesto en 1986 por Rina Dechter (profesora de la Universidad de California, Irvine).

Los seres vivos interpretamos el mundo que nos rodea ejecutando un proceso de alto nivel de abstracción, en múltiples pasos de complejidad creciente. Por ejemplo, para llegar a interpretar que la imagen siguiente muestra “un chico con una gorra y una chica montando un robot con Lego”, los seres humanos llevamos a cabo un proceso en fases, partiendo del análisis de píxeles de la imagen (detección de bordes, colores, etc.) para llevar a cabo la detección y reconocimiento de los objetos presentes, luego se ha analizado la interacción entre ellos para por último ser capaz de generar una descripción de la escena.



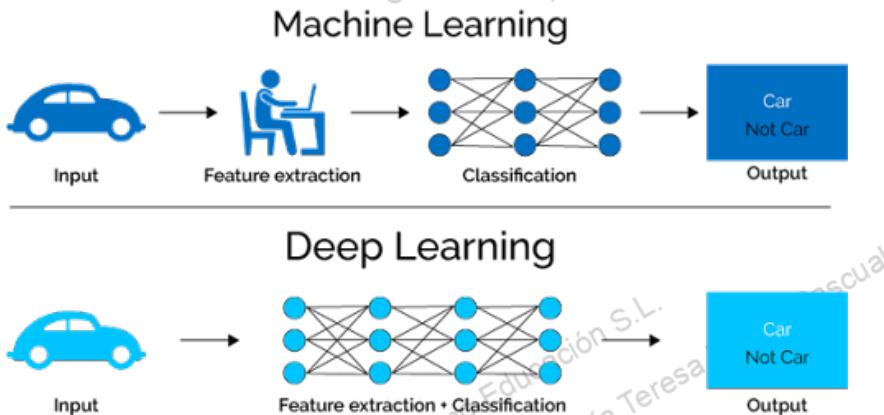
Fuente de la imagen: Stock Photos

Esto tiene lugar en una zona específica de nuestro cerebro dedicada a la visión, dividida en diferentes regiones: el *Lateral Geniculate Nucleus* (LGN) es el centro de procesamiento primario de la información visual recibida por la retina del ojo, y el córtex visual es centro principal de procesamiento de la información visual, dividido también en zonas.



Para saber más sobre el Núcleo Geniculado Lateral, puedes pinchar en este [enlace](#).

La potencia del *deep learning* reside en que se puede construir una red de neuronas con cientos o miles de millones de parámetros, es decir, pesos y conexiones entre neuronas, llegando a parecerse al propio modelo biológico de los seres vivos. Su capacidad es tan elevada que, a diferencia de los algoritmos de aprendizaje automático “clásico”, directamente integra y automatiza la extracción y selección de características (selección de las variables más relevantes para la construcción del modelo, es decir, aquellas que contribuyen con más información para calcular la variable de salida) en el propio proceso de aprendizaje. Es decir, no solo se aprende el modelo o función que relaciona las entradas y las salidas sino que en el mismo proceso se aprende primero qué características tiene que utilizar.



Extracción de características y clasificación de ejemplos

Fuente de la imagen: [Towardsdatascience.com](https://towardsdatascience.com/)

En la fase de entrenamiento, el algoritmo de propagación hacia atrás de errores o retropropagación (en inglés, *backpropagation*) utiliza el error obtenido en la salida y lo va propagando hacia las capas previas ajustando por sí misma el valor de los pesos (minimizando el coste). La importancia del algoritmo recae en la capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados de ejemplo y sus correspondientes salidas.



Para saber más sobre la Propagación hacia atrás, puedes pinchar en este [enlace](#).



El empleo de las técnicas de *deep learning* ha supuesto una revolución en numerosos campos:

- Un nivel casi humano para la clasificación de imágenes.
- Un nivel casi humano para el reconocimiento de la voz.
- Un nivel casi humano en el reconocimiento de escritura (OCR).
- Alcanzado un nivel maestro (superior al humano) en juegos.
- Grandes mejoras en traductores automáticos.
- Grandes mejoras en conversaciones *text to speech*.
- Asistentes digitales como Alexa, Google Assistant o Siri.
- Grandes mejoras en vehículos autónomos.
- Mejores resultados de búsqueda en la web.
- Grandes mejoras para responder preguntas en lenguaje natural.

El gran auge de estas técnicas en los últimos dos años ha sido causado por:

La disponibilidad de conjuntos de datos enormes y de buena calidad

Las técnicas de redes neuronales requieren de un volumen de datos de entrenamiento muy elevado, y más las técnicas de *deep learning*. Hoy en día hay disponibles conjuntos de entrenamiento de millones de ejemplos, de buena calidad, en todos los sectores de actividad.

Computación rápida con GPU

En 1999 se le atribuye a la empresa Nvidia la fabricación de la primera GPU (*Graphics Processing Unit*). Las GPU son procesadores de alta potencia especializados en cálculos matemáticos, con 200 veces más chips que las CPU habituales, que permiten reducir radicalmente los tiempos de entrenamiento de los modelos de *deep learning*.

Arquitecturas y técnicas novedosas para el proceso de aprendizaje

Se han desarrollado arquitecturas de redes neuronales muy novedosas, inspiradas en gran medida en el modelo biológico del cerebro de los seres vivos y en el modelado en múltiples capas de niveles de abstracción. Se trata de arquitecturas avanzadas como redes convolucionales, redes recurrentes, capas de atención, etc.

Desarrollo de plataformas de software

Se han desarrollado numerosos *frameworks* y plataformas en la nube que simplifican el desarrollo de modelos con arquitecturas *deep learning*, como son:

- TensorFlow (Google).
- Keras (sobre TensorFlow).
- PyTorch (Facebook).
- Otros menos populares: Sonnet, MXNet, Swift for TensorFlow, Gluon, DL4J, ONNX, Chainer.

Las redes neuronales profundas están compuestas combinando un gran número de capas diferentes, con funcionalidad diversa y complementaria en muchos casos. Los tipos más conocidos y empleados son las capas de entrada y salida, capa densa (conectando completamente la capa anterior y posterior), de activación, de normalización, de regularización, de mezclado, de cambio de forma (*reshaping*), de mezclado (*merging*), de aplanado (*flatten*), etc.

En los últimos tiempos se han popularizado una serie de arquitecturas de redes neuronales profundas que funcionan de forma óptima en diferentes escenarios, descritas a continuación.

5.2.1. Redes neuronales convolucionales

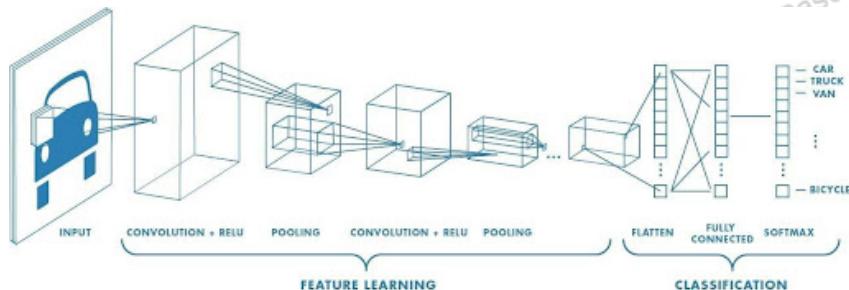
Un tipo muy importante y especial es la capa de **convolución** (*convolution*), que aplica una operación llamada convolución a los datos de entrada (los multiplica con una ventana deslizante basada en una matriz de datos llamada filtro o *kernel*), que aumenta radicalmente la capacidad del modelo. Lo bueno de esta arquitectura es que la red neuronal busca las características más distintivas de los objetos por sí misma. Las primeras capas aprenden a detectar características a bajo nivel, y las posteriores combinan las características anteriores en una representación holística, por lo que a medida que se añaden más capas de convolución, los mapas de características serán capaces de reconocer variables más complejas. Las redes neuronales que incluyen una capa de convolución se llaman redes neuronales convolucionales (*Convolutional Neural Network, CNN*) y su arquitectura de red incluye:

Etapa de detección

Una etapa de detección (*feature learning*), compuesta de una serie de capas convolucionales combinadas con capas de mezclado (*pooling*). Esta reducción (conocida también como *subsampling*) consiste en obtener una muestra de las neuronas más representativas antes de hacer una nueva convolución, preservando así las características más importantes del filtro de convolución. Esto es necesario ya que, si se hiciera directamente una nueva convolución, el número de neuronas capa a capa aumentaría considerablemente, implicando también una mayor carga computacional.

Etapa de clasificación

Una etapa de clasificación (*classification*), que consiste en “aplanar” la salida anterior con una capa “*Flatten*” que se clasifica con una serie de capas densas para generar la salida final.

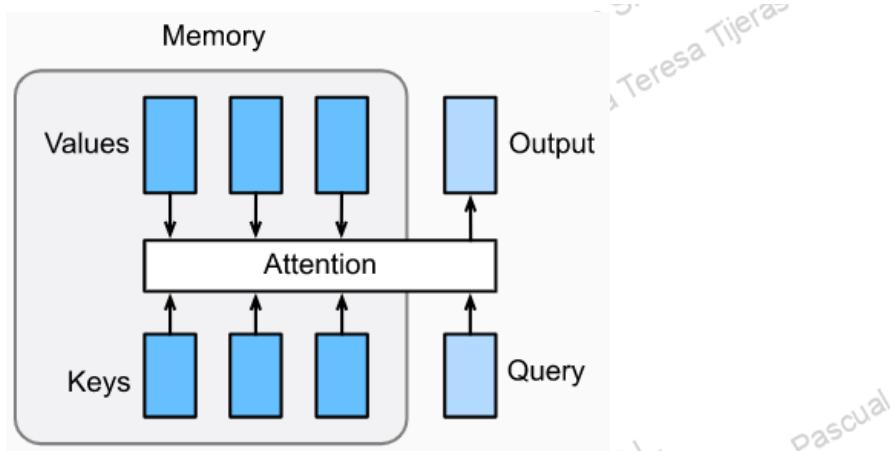


Arquitectura típica de redes neuronales convolucionales

Esta arquitectura se utiliza mucho para clasificación, sobre todo en visión artificial (reconocimiento de objetos, clasificación de imágenes en general), y también para clasificación de texto.

5.2.2. Capa de atención

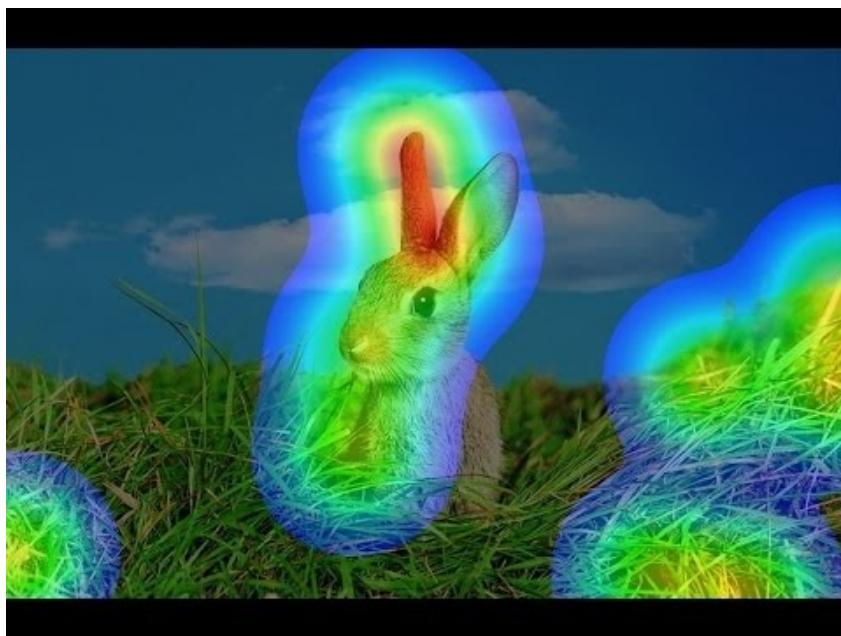
Otra capa muy importante en los modelos de *deep learning* actuales es la capa de **atención** (*attention*), que es un mecanismo diseñado para hacer que la red se centre en los estados más importantes y deseche información irrelevante, almacenando la información más importante del contexto.



Arquitectura de capa de atención, con bloques de memoria

Se asume que las capas de atención han supuesto grandes avances en las tecnologías de la visión artificial y del procesamiento de texto, y son los fundamentos de las arquitecturas más modernas de *deep learning*, como se describe en el modelo de *transformer*.

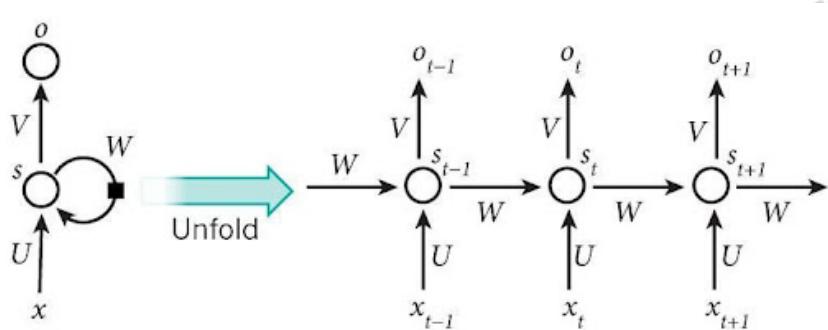
La siguiente figura muestra un ejemplo donde se marcan las áreas de la imagen que la capa de atención detecta como más importantes para el reconocimiento de objetos en la imagen. Así, la red neuronal puede enfocarse en esas áreas para identificar los objetos “*herba*” y “*conejo*”.



Ejemplo de atención en reconocimiento de objetos en visión artificial

5.2.3. Redes neuronales recurrentes

Una red neuronal recurrente (*Recurrent Neural Network*, RNN) es una red neuronal que incorpora un tipo especial de capa llamada recurrente (*recurrent*), que consiste en incluir una realimentación de la salida hacia la entrada (es decir, que la salida de la neurona se conecta de nuevo a su entrada). La figura siguiente muestra la representación típica.



Representación de las redes neuronales recurrentes

Los modelos secuenciales son claves en el procesamiento del lenguaje natural, donde la secuencia y el orden de las palabras resultan cruciales. Ejemplos de sus aplicaciones son el reconocimiento de voz, los sistemas de traducción o el análisis de sentimiento. Las variantes de redes neuronales recurrentes que más se utilizan en procesamiento del lenguaje natural son aquellas que incorporan un mecanismo de atención:

Long-Short Term Memory

Long-Short Term Memory (LSTM) que permiten mantener información a largo plazo (útil en caso de que se requiera un gran contexto).

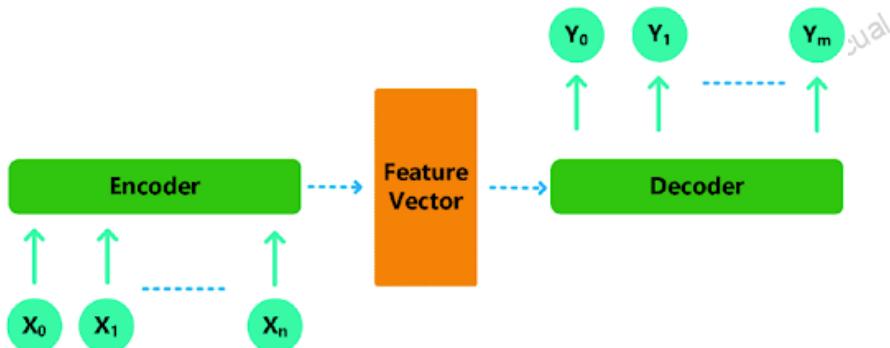
Gated Recurrent Unit

Gated Recurrent Unit (GRU) que mantiene un registro de la información más relevante para la predicción. Básicamente, se determina cuánta información anterior se transmite en el siguiente paso, y cuánta información deja de ser relevante y debe olvidarse.

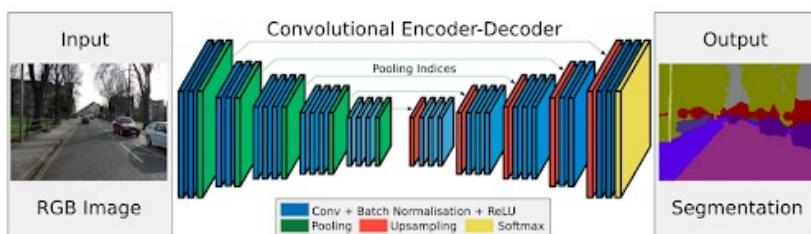
5.2.4. Arquitectura *encoder-decoder*

Este modelo también se denomina *sequence-to-sequence*, o Seq2Seq, porque su objetivo es transformar una secuencia de entrada en otra de salida. Para ello, utiliza una RNN básica o, más a menudo, una LSTM o una GRU.

Su arquitectura consiste en establecer una arquitectura de red neuronal dividida en dos partes: un codificador (*encoder*) que es capaz de capturar la información más importante de los datos de entrada (convierte cada elemento de los datos de entrada en un vector oculto correspondiente que contiene el elemento y su contexto), y un decodificador (*decoder*) que a partir de esa información es capaz de generar una información de salida adaptada a la tarea que se considere (invierte el proceso de codificación, convirtiendo el vector en un elemento de salida, utilizando la salida anterior como contexto de entrada).

*Arquitectura encoder-decoder*

Tiene amplias aplicaciones en visión artificial, por ejemplo, para reconocimiento de objetos y segmentación de imágenes (identificación de los diferentes objetos dentro de la imagen).

*Aplicación de encoder-decoder para segmentación de imágenes*

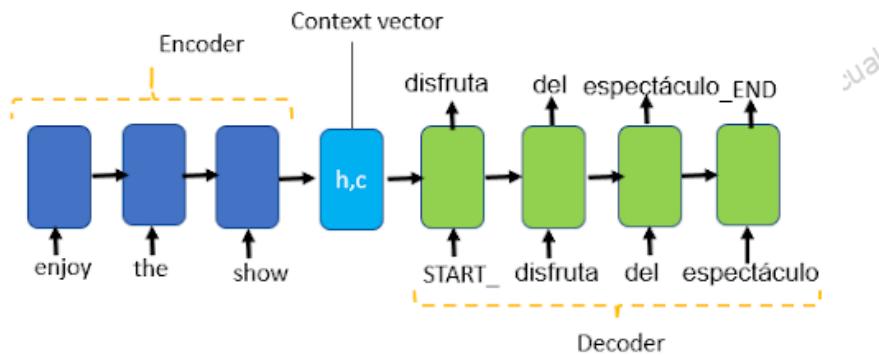
También tiene grandes aplicaciones en el procesamiento del lenguaje natural como, por ejemplo, para traducción automática o generación de texto.

Encoder

El *encoder* captura la información semántica del texto de entrada, detectando las palabras más importantes (aquellas que dan el mayor sentido al texto) y las relaciones entre ellas (sintaxis, contextos léxicos, etc.) y almacenando esta información en los mecanismos de atención incorporados.

Decoder

El *decoder* realiza la transformación de la secuencia en la salida esperada, por ejemplo, en el caso de traducción automática, consistiría en traducir la secuencia de entrada en un idioma origen (por ejemplo, inglés) en una secuencia de salida en el idioma destino (por ejemplo, en francés). Para generar cada palabra utiliza la información capturada por el codificador y los diferentes estados almacenados por las capas de atención del modelo.



Aplicación de encoder-decoder para traducción automática
Fuente de la imagen: [Towardsdatascience.com](https://towardsdatascience.com/encoder-decoder-for-machine-translation-101-10f3a2a2a2d0)

5.2.5. Modelo de *transformer*

Este modelo fue propuesto por un equipo de investigación de Google en 2017 y supuso una verdadera revolución en el procesamiento del lenguaje natural utilizando *deep learning* [Vaswani et al. (2017). *Attention is All You Need*].

Antes de ellos, la mayoría de los sistemas se basaban en redes neuronales recurrentes (RNN) que incorporaban mecanismos de atención, como LSTM y GRU. Los *transformers* son una variante del modelo Seq2Seq que incluye múltiples capas de atención y *embeddings*, sin utilizar una estructura RNN, lo que pone de manifiesto que los mecanismos de atención por sí solos pueden igualar el rendimiento de las RNN con atención. Los *transformers* se adaptan a cualquier tarea de lenguaje natural: clasificación, traducción automática, similitud semántica, respuesta a preguntas, etc.

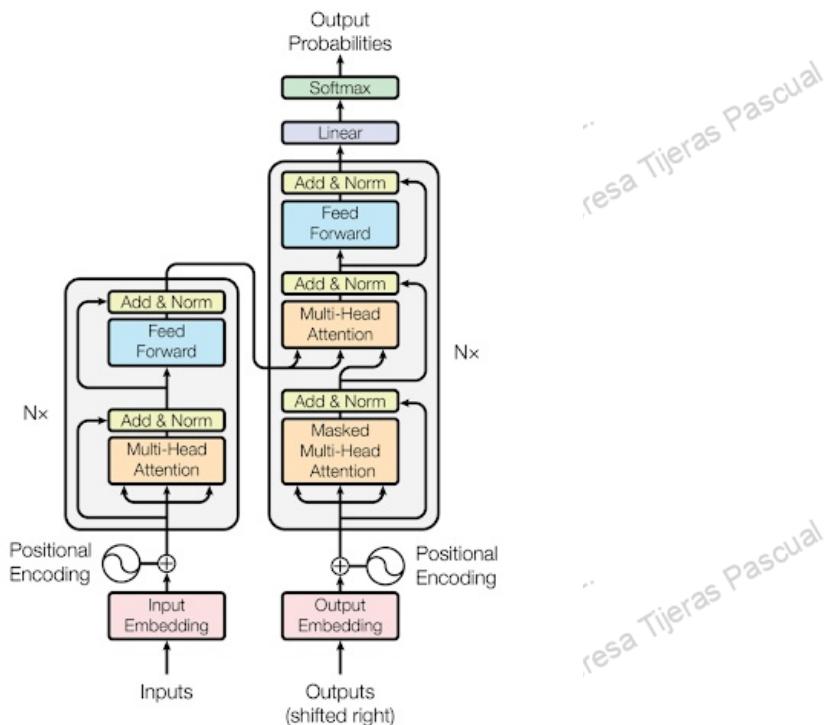


Figure 1: The Transformer - model architecture.

Arquitectura de transformer

Fuente de la imagen: Vaswani et al. (2017). *Attention is All You Need*

Con *transformers* se suele trabajar en dos fases:

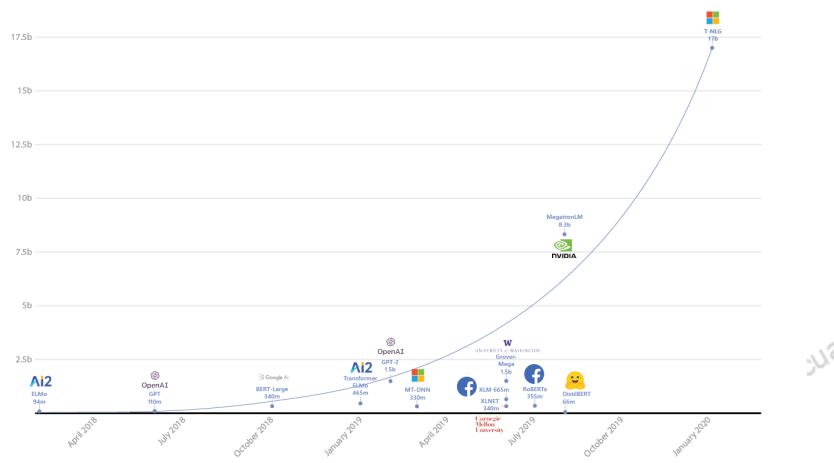
Pre-training

El modelo aprende cómo se estructura el lenguaje de forma general.

Fine-tuning

Se añaden capas adicionales para adaptar el modelo a tareas concretas.

Estos modelos han avanzado mucho gracias a los avances tecnológicos y a los grandes volúmenes de texto en diferentes idiomas disponibles gratuitamente en Internet. Además, la obtención del mejor modelo se ha convertido en una competición entre los grandes jugadores del mercado. Algunos de los principales son los modelos de BERT, de Google, o los GPT de OpenAI.



Evolución de los grandes modelos de PLN

Fuente de la imagen: [Kdnuggets.com](https://www.kdnuggets.com/2019/07/evolution-large-nlp-models.html)



Para saber más sobre el *transformer*, puedes pinchar en [este](#) y [este](#) otro enlace.

5.2.6 Ventajas e inconvenientes

Pese a sus grandes ventajas, los problemas más importantes de las técnicas de *deep learning* son las siguientes:

1

Requiere de una cantidad ingente de datos de entrenamiento.

2

Computacionalmente muy intensivo.

3

Complejo de entender, requiere expertos. Arquitecturas de “caja negra” (modelos matemáticos complejos), difícil de interpretar, no se pueden ajustar de forma directa (como los diccionarios o las reglas).

4

Puede sufrir de “sesgo algorítmico”. El sesgo algorítmico (*algorithmic bias*) consiste en la presencia de errores sistemáticos y repetitivos en un sistema informático que crean resultados injustos o políticamente incorrectos. El sesgo puede surgir debido a muchos factores, incluidos, entre otros, el diseño del algoritmo o el uso o las decisiones no intencionadas o imprevistas relacionadas con la forma en que se escogen, seleccionan, codifican o utilizan los datos para entrenar el algoritmo.

Por ejemplo, en 2015, Google tuvo que pedir disculpas cuando los usuarios de raza negra se quejaron de que un algoritmo de identificación de imágenes de su aplicación Google Photos los identificaba como gorilas. En 2010, hubo muchas críticas sobre las cámaras Nikon, ya que los algoritmos de reconocimiento de imágenes preguntaban sistemáticamente a los usuarios asiáticos si habían parpadeado en el momento de hacer la foto.

Otro caso real de los primeros sistemas de análisis de sentimiento basados en *deep learning* es el siguiente:

Text: i'm a dog

Sentiment: 0.0

Text: i'm a homosexual

Sentiment: -0.5

Text: i'm a homosexual dog

Sentiment: -0.600000238418579

Ejemplo de sesgo algorítmico en análisis de sentimientos

Puede sufrir “*adversarial attacks*”. El “*adversarial attack*” es una técnica que intenta engañar a los sistemas con datos engañosos, tratando de confundir su salida. Para ello utiliza un tipo de técnica que se llama “aprendizaje automático adversario”.

Por ejemplo, en 2019, McAfee atacó el antiguo sistema Mobileye de Tesla, engañándolo para que condujera a 80 km/h por encima del límite de velocidad, simplemente añadiendo una tira de cinco centímetros de cinta negra a una señal de límite de velocidad. Otros ataques consisten en proyectar imágenes de personas o señales sobre la carretera para confundir al sistema de conducción autónoma.



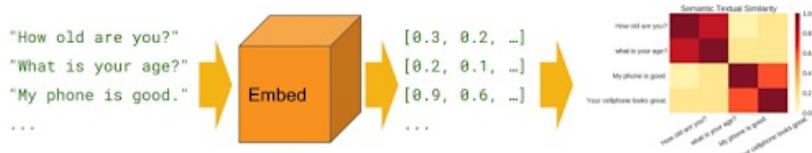
Ejemplo de “adversarial attack” a vehículos Tesla

5.3. Vectores de *embeddings*

Los vectores de “*embeddings*” (término difícil de traducir al español, sería algo como “incrustaciones”) es el nombre de un conjunto de lenguajes de modelado y técnicas de aprendizaje en procesamiento del lenguaje natural, surgido en torno a 2013, según los cuales las palabras o frases de un texto natural se representan como vectores de números reales.

Estos vectores codifican el significado de cada palabra en el corpus utilizado como entrenamiento, de tal forma que las palabras cuyos vectores estén más cercanos en el espacio vectorial se supone que tienen un significado similar (son sinónimos). Por tanto, es posible llevar a cabo operaciones de similitud semántica basándose en operaciones sobre la distancia entre vectores: las palabras “más parecidas” serán aquellas cuya distancia entre vectores sea menor. Es decir, dado un conjunto de palabras, es posible encontrar cuáles son sinónimos simplemente agrupando por las menores distancias entre vectores.

La siguiente figura muestra un ejemplo de varias frases. A cada una de ellas, le corresponde un vector de *embeddings*, que se calcula como la suma de los vectores de *embeddings* de las diferentes palabras que componen la frase. Para encontrar qué frases tienen un significado similar solo es necesario calcular la distancia euclídea entre los vectores y quedarse con aquellas frases cuya distancia sea menor a un umbral dado (como “How old are you” y “What is your age”, en el ejemplo).



Ejemplo de similitud semántica

Además, otra propiedad interesante es que se puede aplicar la aritmética de composición de vectores. Es un ejemplo clásico de los primeros modelos de *embeddings* que realizando operaciones sobre vectores es posible llegar desde una palabra a otra. Por ejemplo, tomando el vector de la palabra "king" (rey), restándole el vector de la palabra "man" (hombre), por ejemplo, mediante la suma euclídea componente a componente, y sumándole el vector de "woman" (mujer), obtenemos el vector de la palabra "queen" (reina). Es decir:



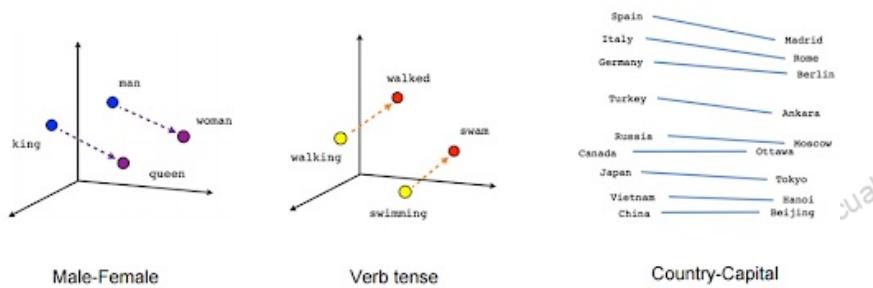
$$\text{vec(king)} - \text{vec(man)} + \text{vec(woman)} = \text{vec(queen)}$$

De forma similar sucede con:



$$\text{vec(walking)} - \text{vec(walk)} + \text{vec(swim)} = \text{vec(swimming)}$$

$$\text{vec(Madrid)} - \text{vec(Spain)} + \text{vec(France)} = \text{vec(Paris)}$$

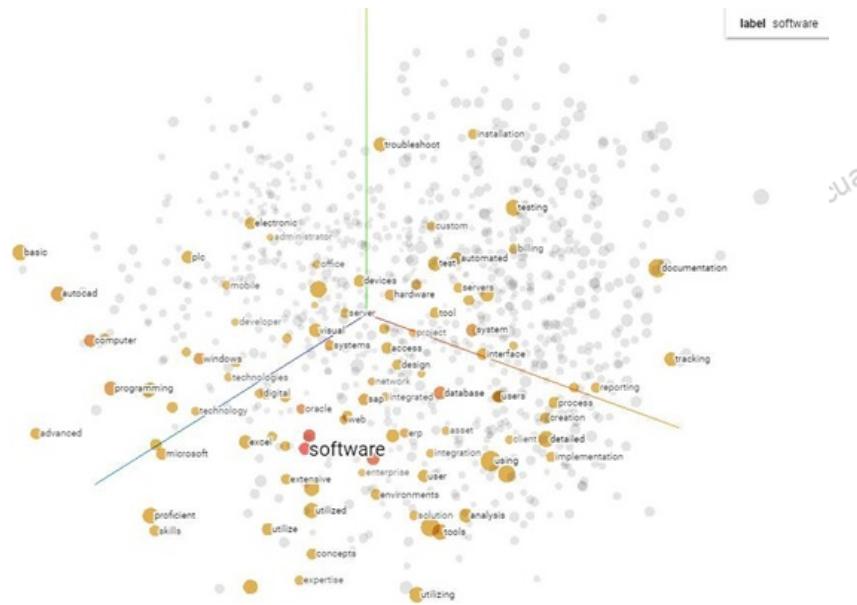


Similitud semántica y operaciones entre vectores

Está ampliamente demostrado que los modelos de *embeddings* (para palabras individuales o frases completas), cuando se utilizan como representación de la entrada, aumentan el rendimiento en casi todas las tareas de procesamiento del lenguaje natural, en particular aquellas que se basan de una forma u otra en la similitud semántica.

Existen diferentes métodos de generar estos vectores de *embeddings*, como modelos probabilísticos, redes neuronales, métodos de reducción de dimensionalidad en la matriz de coocurrencia de palabras, etc. Los primeros modelos de *embeddings* eran independientes del contexto (cada palabra se trataba de forma aislada a las demás, lo que resulta un problema en las palabras polisémicas), aunque ahora, con el uso de arquitecturas Seq2Seq, los modelos de *embeddings* son contextuales y son capaces de diferenciar entre los diferentes significados de una palabra polisémica.

Aunque era posible entrenar modelos propios, diferentes organizaciones han publicado modelos de *embeddings* entrenados con millones de palabras que eran útiles para contextos generales (por ejemplo, usando todo el contenido de Wikipedia).



Ejemplo de representación de la palabra “software” en el espacio de vectores de embeddings



Los modelos iniciales de *embeddings* independientes de contexto más conocidos fueron:

- Word2vec (Google).
- GloVe (Stanford).
- Fasttext (Facebook).



Los modelos actuales (basados en *deep learning* con arquitecturas Seq2Seq) más populares (y que han sustituido casi completamente a los anteriores) son:

- BERT (*Bidirectional Encoder Representations from Transformers*), de Google, y sus infinitas variantes: RoBERTa, DistilBERT, ALBERTa...
- GPT (OpenAI).
- Otros: XLM, XLNet, T5, CTRL.

El aprendizaje por transferencia (*transfer learning*) es muy importante en *deep learning*. Consiste en entrenar un modelo sobre un conjunto de datos a gran escala y luego usar ese modelo (previamente entrenado) para aprender otra tarea posterior (es decir, la tarea objetivo).

Aplicar el conocimiento ya adquirido por un modelo ayuda a reducir el tiempo de entrenamiento de nuevos modelos, ya que permite a los sistemas no partir de cero y así no se requiere de conjuntos de datos tan voluminosos como los necesarios para desarrollar un modelo desde el inicio.

Los avances recientes en la investigación combinan los métodos de aprendizaje por transferencia con los modelos de lenguaje basados en arquitecturas de *transformers* consiguiendo modelos de propósito general. Estos modelos preentrenados, como BERT, permiten a los expertos focalizarse en su objetivo sin tener que construir el modelo desde cero y reduciendo el volumen de texto necesario.

Se podría decir que el modelo ya entiende el lenguaje y solo hay que enseñarle a resolver la tarea específica.

Cada día se publican nuevos modelos preentrenados para dominios generales o específicos (salud, finanzas, seguridad informática, etc.), para un único idioma o representaciones multilingües (modelos que sirven a la vez para diferentes idiomas).



La implementación más conocida del modelo de *transformers* es [HuggingFace](#), que aparte de proporcionar bibliotecas de código Python para tareas de procesamiento del lenguaje natural empleando *deep learning*, también es un repositorio de modelos preentrenados por la comunidad para diferentes tareas. Hoy por hoy disponen de casi [14.000 modelos](#).

A screenshot of the HuggingFace website's model repository page. The top navigation bar includes 'Hugging Face', a search bar, and links for 'Models', 'Datasets', 'Resources', 'Solutions', 'Pricing', 'Log In', and 'Sign Up'. The main area is divided into sections: 'Tasks' (Fill-Mask, Question Answering, Summarization, Table Question Answering, Text Classification, Text Generation, Text2Text Generation, Token Classification, Translation, Zero-Shot Classification, Sentence Similarity), 'Libraries' (PyTorch, TensorFlow, JAX), 'Datasets' (wikipedia, contl2003, common_voice, dcepc_europarl_jrc-acquis, squad, oscar, bookcorpus, CLUECorpusSmall), 'Languages' (en, es, fr, de, sv, fi, zh, ru), and 'Licenses' (apache-2.0, mit, cc-by-4.0). The central column shows a grid of pre-trained models with their names, descriptions, and download counts. Some examples include 'bert-base-uncased', 'roberta-large', 'distilbert-base-uncased-finetuned-sst-2-english', 'sentence-transformers/paraphrase-xlm-rand1024', 'xlm-roberta-base', 'google/bert_uncased_L-12_H-512_A-8', 'google/mt5-base', 'deepset/roberta-base-squad2', 'bert-large-uncased-whole-word-masking-fin', 'distilbert-base-uncased', 'bert-base-cased', 'roberta-base', 't5-base', 'finiteautomata/beto-sentiment-analysis', and 'gpt2'. A sidebar on the right allows sorting by 'Most Downloads'.

Repositorio de modelos de HuggingFace



RESUMEN

En esta unidad se ha presentado una panorámica general de las tareas del procesamiento del lenguaje natural dedicadas a la **extracción de información** estructurada a partir de información no estructurada, es decir, textos.

En las diferentes tareas presentadas se ve que la extracción de información puede realizarse con **diferentes niveles de abstracción y complejidad variable**, desde la más sencilla extracción de entidades, pasando por el reconocimiento de información compleja en forma de grafos semánticos de objetos y relaciones entre ellos, la clasificación automática, así como otras tareas como recuperación de información, respuesta a preguntas o extracción de resúmenes.

Además, se han presentado una serie de **métricas generales** que sirven para la evaluación de este tipo de sistemas. Más adelante, en futuras unidades del curso, se describirán las métricas particulares para cada una de las tareas, en muchos casos basadas en estas métricas generales.

Por último, se han presentado los conceptos principales de **aprendizaje automático**, como técnica ampliamente utilizada en inteligencia artificial en general y procesamiento del lenguaje natural en particular. **Deep learning** (o aprendizaje profundo) es un tipo específico de aprendizaje automático que se basa en modelos de redes neuronales con un elevado número de capas (de ahí lo de “profundo”), que se inspiran en el modelo biológico del sistema nervioso central de los seres vivos. Este enfoque ha logrado una potencia computacional tan elevada que con su aplicación la máquina es ya capaz de abordar tareas de alta complejidad similar a las que llevamos a cabo los humanos. Además, se han descrito las técnicas de representación semántica utilizando **embeddings**, tan populares hoy en día.

Cálamo Educación S.L.
María Teresa Tijeras Pascual
-ual

Recursos

Enlaces de Interés



What is Information Extraction?

<https://www.ontotext.com/knowledgehub/fundamentals/information-extraction/>



Message Understanding Conferences

https://en.wikipedia.org/wiki/Message_Understanding_Conference



Introduction to Information Extraction

https://www-nlpir.nist.gov/related_projects/muc/



Enterprise Information Portals

https://www.researchgate.net/publication/28803762_Enterprise_information_portal_a_new_paradigm_in_resource_discovery



Structured Data vs. Unstructured Data: what are they and why care?

<https://lawtomated.com/structured-data-vs-unstructured-data-what-are-they-and-why-care/>



Pablo Picasso

https://es.wikipedia.org/wiki/Pablo_Picasso



Ontology components

https://en.wikipedia.org/wiki/Ontology_components



Text Classification, Google

<https://developers.google.com/machine-learning/guides/text-classification>



Google's Knowledge Graph Explained: How It Influences SEO

<https://ahrefs.com/blog/google-knowledge-graph/>



Resource Description Framework

https://es.wikipedia.org/wiki/Resource_Description_Framework



Turtle (syntax)

[https://en.wikipedia.org/wiki/Turtle_\(syntax\)](https://en.wikipedia.org/wiki/Turtle_(syntax))



TREC Logo

<https://www.nist.gov/image/newlogopng>



Building a Question-Answering System from Scratch - Part 1

<https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507>



Paraphrasing, quoting and summarising: Summary example

<https://libguides.newcastle.edu.au/paraphrasing-summarising/example-of-summarising>



What is the typical architecture of an AI chatbot?

<https://www.quora.com/What-is-the-typical-architecture-of-an-AI-chatbot>



Write With Transformer

<https://transformer.huggingface.co/>



Talk To Transformers

<https://app.inferkit.com/demo>

-  **GPT-2**
<https://en.wikipedia.org/wiki/GPT-2>
-  **GPT-3**
<https://en.wikipedia.org/wiki/GPT-3>
-  **Modelos de EleutherAI**
<https://huggingface.co/EleutherAI>
-  **Confusion matrix**
https://en.wikipedia.org/wiki/Confusion_matrix
-  **Precisión y recuperación (Precision and recall)**
https://medium.com/@gogasca/_precisi%C3%B3n-y-recuperaci%C3%B3n-precision-recall-dc3c92178d5b
-  **Precision and recall**
https://en.wikipedia.org/wiki/Precision_and_recall
-  **La matriz de confusión y sus métricas**
<https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>
-  **Performance Metrics for Text Categorization**
<https://www.meaningcloud.com/blog/performance-metrics-for-text-categorization>
-  **Tipos de aprendizaje automático**
<https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2>
-  **Neurona**
<https://es.wikipedia.org/wiki/Neurona>
-  **Introduction to Deep Learning**
<https://www.kdnuggets.com/2018/09/introduction-deep-learning.html>
-  **Núcleo Geniculado Lateral**
https://es.wikipedia.org/wiki/N%C3%BCcleo_geniculado_lateral
-  **What makes Deep Learning different from traditional Machine Learning methods?**
<https://towardsdatascience.com/what-makes-deep-learning-different-from-traditional-machine-learning-methods-fc154db33939>
-  **Propagación hacia atrás**
https://es.wikipedia.org/wiki/Propagaci%C3%B3n_hacia_atr%C3%A1s
-  **TensorFlow**
<https://www.tensorflow.org/>
-  **Keras**
<https://keras.io/>
-  **PyTorch**
<https://pytorch.org/>
-  **Implementing neural machine translation using keras**
<https://towardsdatascience.com/implementing-neural-machine-translation-using-keras-8312e4844eb8>
-  **Attention is All You Need**
<https://arxiv.org/abs/1706.03762>

-  **[Microsoft Open Sources ZeRO and DeepSpeed: The Technologies Behind the Biggest Language Model in History](https://www.kdnuggets.com/2020/02/microsoft-open-sources-zero-deepspeed-language-model.html)**
<https://www.kdnuggets.com/2020/02/microsoft-open-sources-zero-deepspeed-language-model.html>
-  **Transformer**
[https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model))
-  **How Transformers work in deep learning and NLP: an intuitive introduction**
<https://theaisummer.com/transformer/>
-  **HuggingFace**
<https://huggingface.co/>
-  **Modelos de transformers en HuggingFace**
<https://huggingface.co/models>