



MODELOS DE I.A.

<https://martinezpenya.es/ModelosIA/>
© 2025 David Martínez licensed under CC BY-NC-SA 4.0

Modelos de IA (CE Inteligencia Artificial y Big Data)

IES Eduardo Primo Marques (Carlet)

David Martinez Peña

© 2025 David Martínez licensed under CC BY-NC-SA 4.0

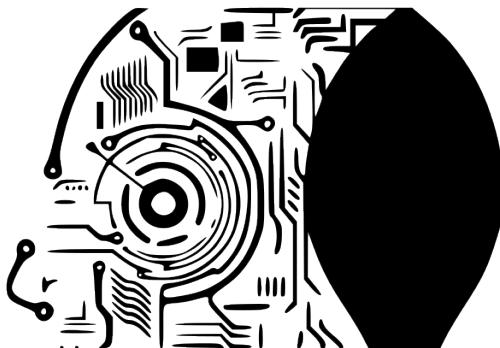
Table of contents

1. UD00	4
1.1  Información importante	4
2. UD01	6
2.1 Caracterización de sistemas y utilización de modelos de Inteligencia Artificial	6
2.2 Diapositivas de la UD01	44
2.3 Actividades	45
2.4 Practicas	65
3. UD02	78
3.1 Inteligencia Artificial Simbólica	78
4. UD03	0
4.1 Introducción	0
4.2 El procesamiento del lenguaje natural estudia las relaciones del lenguaje entre los seres humanos y las máquinas.	0
4.3 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.	0
4.4 Técnicas de procesamiento de lenguaje. Limitaciones	0
4.5 Formación del investigador en PLN	0
4.6 Elaboración de un sistema de procesamiento de lenguaje orientado a una tarea específica	0
4.7 Fuentes de información	0
5. UD04	0
5.1 Robots	0
5.2 Hardware de Robots	0
5.3 ¿Qué tipo de problema resuelve la robótica?	0
5.4 Percepción robótica	0
5.5 Planificación y Control	0
5.6 Planificación de movimientos inciertos	0
5.7 Aprendizaje por refuerzo en robótica	0
5.8 Humanos y robots	0
5.9 Dominios de aplicación	0
5.10 Resumen	0
5.11 Fuentes de información	0
6. UD05	0
6.1 Introducción	0
6.2 Deontología profesional en inteligencia artificial	0
6.3 La privacidad de los datos	0
6.4 Protección frente a errores	0

6.5 Principios éticos	0
6.6 Sesgos en el desarrollo y aplicación de la IA y el Big Data	0
6.7 Regulación Europea sobre IA	0
6.8 Fuentes de información	0
7.  Sobre mí...	0
7.1  David Martínez Peña	0
7.2  Contacto:	0
8. Fuentes de información	0

1. UD00

1.1 🚀 Información importante



MODELOS DE I.A.

<https://martinezpenya.es/ModelosIA/>

© 2025 David Martínez licensed under CC BY-NC-SA 4.0

📅 Denominación del curso

📚 Curso de especialización de Inteligencia Artificial y Big Data

🤖 Modelos de Inteligencia Artificial (MIA)

📋 Contenidos

Bloque	TRIMESTRE/UNIDAD	Horas
	PDF 17 PRIMER TRIMESTRE	44
1AVA	❖ UD01: Caracterización de sistemas y utilización de modelos de Inteligencia Artificial	27
	🧠 UD02: Sistemas Expertos	15
📊 1a EVALUACIÓN		2
	PDF 17 SEGUNDO TRIMESTRE	46
2AVA	🗣 UD03: Procesamiento del Lenguaje Natural	15
	🤖 UD04: Análisis de sistemas robotizados	15
	⚖ UD05: Aplicación de principios legales y éticos de la Inteligencia Artificial	14
📊 2ª EVALUACIÓN		2
🎓 CONVOCATÒRIA ORDINÀRIA		2
	📊 TOTAL	90

🎯 Resultados de Aprendizaje (RA)

	Descripción	Unidades	Nota mínima	Peso
RA1	🔍 Caracteraiza sistemas de Inteligencia Artificial relacionándolos con la mejora de la eficiencia operativa de las organizaciones y empresas.	1	5	10,00%
RA2		1	5	10,00%

	Descripción	Unidades	Nota mínima	Peso
	⚙ Utiliza modelos de sistemas de Inteligencia Artificial implementando sistemas de resolución de problemas.			
RA3	💬 Relaciona el procesamiento de lenguaje natural con sus aplicaciones determinando su potencial e identificando sus limitaciones.	3	5	20,00%
RA4	🤖 Analiza sistemas robotizados, evaluando opciones de diseño e implementación.	4	5	20,00%
RA5	🧠 Aplica sistemas expertos evaluando la influencia de los controladores inteligentes en el comportamiento del sistema.	2	5	20,00%
RA6	⚖️ Aplica principios legales y éticos al desarrollo de la Inteligencia Artificial integrándolos como parte del proceso.	5	5	20,00%
100,00%				

Legislación vigente

- [RD 497/2024 21-05-2024](#)
- [RD 279/2021 20-04-2021](#)
- [Horario](#)

Evaluación

- 🔎 La evaluación del módulo se realizará con base en los **Resultados de Aprendizaje (RA)** definidos en el currículo del Curso de especialización de Inteligencia Artificial y Big Data. Cada RA estará asociado a **criterios de evaluación (CE)** que serán los que determinen el grado de adquisición de las competencias previstas para el módulo.
- 📊 La nota final del módulo se obtendrá a partir de la ponderación de los **RA**, como se mencionó anteriormente. Cada **RA** será evaluado de forma independiente, con calificaciones en una escala de 0 a 10.
- ✅ El alumno debe obtener al menos una nota de **5** en cada **RA** para aprobar el módulo.
- ↗ Si un alumno obtiene menos de un **5** en algún RA, tendrá que recuperarlo mediante las actividades/exámenes de recuperación diseñadas específicamente para esos resultados de aprendizaje.

⚠ IMPORTANTE:

- ! Aprobar las distintas evaluaciones no garantiza aprobar el curso.
- ❤️ Puedes aprobar (y con muy buena nota) las dos evaluaciones, tener un **RA** suspendido y por tanto suspender el módulo.

⌚14 de julio de 2025

2. UD01

2.1 Caracterización de sistemas y utilización de modelos de Inteligencia Artificial

Fundamentos de los Sistemas Inteligentes

Definición de Inteligencia Artificial (IA)

La Inteligencia Artificial es un campo de la informática y la ciencia de la computación que se enfoca en la creación de sistemas y programas capaces de realizar tareas que normalmente requieren inteligencia humana.

Una primera definición bastante común que podemos encontrar para la Inteligencia Artificial es:

"Habilidad para aprender y resolver problemas, llevada a cabo por una máquina o software"

En general, la mayoría de los expertos coinciden en que es la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Estos procesos incluyen:

1. El **aprendizaje** a través de la adquisición de información y reglas para el uso de la información.
2. El **razonamiento** usando las reglas para llegar a conclusiones aproximadas o definitivas.
3. La **autocorrección**.

Una definición más concreta y consensuada podría ser:

"La inteligencia artificial es la inteligencia llevada a cabo por máquinas. En ciencias de la computación, una máquina «inteligente» ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea".

En realidad, cada generación de hardware y software ha asignado este término a las arquitecturas y técnicas de vanguardia en ese momento. Es por esto que la propia definición puede ir cambiando y evolucionando a medida que se van alcanzando metas más ambiciosas. Podríamos decir que cada nueva oleada de avance tecnológico en este ámbito pasa a conformar una nueva definición de inteligencia artificial, o, al menos, añade un matiz propio a ésta.

La realidad actual es que la IA despierta tanta fascinación como desconfianza. En la gran mayoría de los casos, no se conoce bien la técnica con la que se desarrolla. Ese desconocimiento es el que favorece que se mezcle la realidad con las influencias y fantasías de lo leído y visto en novelas, series y películas.

¿Qué es realmente posible con la tecnología actual y qué sigue perteneciendo al campo de la ciencia ficción? Una habilidad que debe tener cualquier profesional del campo de la IA es, precisamente, ser capaz de explicar de forma sencilla las verdaderas amenazas que esta tecnología puede representar, y saber transmitir una imagen de responsabilidad al respecto.

Los ordenadores (y con ellos la inteligencia artificial) no son ni buenos ni malos. Hacen lo que los humanos programamos que hagan.

La inteligencia y, sobre todo, la intencionalidad que pueda tener un programa o aplicación la proporciona el humano (o equipo de humanos) que lo definen y desarrollan.

Historia de la IA

A lo largo de la historia, la IA ha pasado por diferentes etapas de desarrollo, con avances y desafíos significativos.

Prehistoria de la IA o proto-IA (Antes del 1950): En realidad, antes de 1956 ya hubo una serie de hitos científicos que podríamos considerar parte del nacimiento de la Inteligencia Artificial:

En 1943 McCulloch y Pitts presentaron un primer modelo de lo que podría ser una neurona artificial, publicándose en el Boletín de Biofísica Matemática con el título: "A logical calculus of the ideas immanent in nervous activity". Partieron de tres fuentes: conocimientos sobre la fisiología básica y funcionamiento de las neuronas en el cerebro, el análisis formal de la lógica preposicional de Russell y Whitehead y la teoría de la computación de Turing. Es por esto que se consideran los eventos más importantes en el origen de la IA (Russell, S., y Norvig, P. 2008)

En 1950, en el trabajo "Computing Machinery and Intelligence", Alan Turing define la conducta inteligente de la máquina como la capacidad de lograr eficiencia a nivel humano en todas las actividades de tipo cognoscitivo, suficiente para engañar a un evaluador humano, y da forma al famoso "Test de Turing". En este histórico artículo Turing propuso que la pregunta «¿puede pensar una máquina?» era demasiado filosófica para tener valor y, para hacerlo más concreto, propuso un «juego de imitación».

En la prueba de Turing intervienen dos personas y una computadora. Una persona, el interrogador, se sienta en una sala y teclea preguntas en la terminal de una computadora. Cuando aparecen las respuestas en la terminal, el interrogador intenta determinar si fueron hechas por otra persona o por una computadora. Si la computadora actúa de manera inteligente, según Turing, es inteligente. Turing continúa, "Ahora podemos preguntar: '¿Qué sucederá cuando una máquina tome el papel de A en este juego?' ¿Decidirá el interrogador erróneamente tan a menudo cuando se juegue de esta manera como lo hace cuando el juego se juega entre un hombre y una mujer? Estas preguntas reemplazan nuestra pregunta original: '¿Pueden las máquinas pensar?' "(Turing, 1950) La configuración de la prueba puede representarse de esta manera:

```
flowchart LR
    C[Interrogador]
    A[A: Hombre]
    B[B: Mujer]
    D[FIN]
    C-- Hace una pregunta --> A
    A-- Responde --> C
    C-- Hace una pregunta --> B
    B-- Responde --> C
    C-- Decide si A/B es humano --> D
```

Este test puede servir, como señala Turing, no solo para probar una destreza verbal superficial, sino también el conocimiento de fondo y la capacidad de razonamiento subyacente, ya que los interrogadores pueden hacer cualquier pregunta o plantear cualquier desafío verbal que elijan. Con respecto a este test, Turing predijo famosamente que "*dentro de unos cincuenta años [para el año 2000] será posible programar computadoras... para hacer que jueguen el juego de imitación tan bien que un interrogador promedio tendrá no más del 70 por ciento de probabilidad de hacer la identificación correcta después de cinco minutos de interrogación*" (Turing 1950); una predicción que ha fallado notoriamente. En el año 2000, las máquinas en la competición del Premio Loebner jugaron tan mal el juego que el interrogador promedio tuvo un 100 por ciento de probabilidad de hacer la identificación correcta después de cinco minutos de interrogación (ver Moor 2001).

Primeros Conceptos de IA (Décadas de 1950 y 1960): El término "Inteligencia Artificial" fue acuñado por John McCarthy en 1956, durante una conferencia (Dartmouth Summer Research Conference on Artificial Intelligence) que se considera el punto de inicio oficial del campo. En dicho encuentro, John McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon, Ray Solomonoff, Oliver Selfridge, Trenchard More, Arthur Samuel, Herbert Simon y Allen Newell, crearon la conjectura inicial "*Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it*". En esa época, los investigadores estaban entusiasmados por la idea de que las computadoras pudieran ser programadas para simular la capacidad de razonar y resolver problemas como lo hace el ser humano. Se realizaron esfuerzos iniciales para desarrollar programas que pudieran jugar al ajedrez, realizar cálculos matemáticos y comprender lenguaje natural. En esta conferencia se hicieron previsiones triunfalistas a diez años que jamás se cumplieron (como diríamos ahora "**se vinieron muy arriba**").

1956 Dartmouth Conference: The Founding Fathers of AI



John McCarthy



Marvin Minsky



Claude Shannon



Ray Solomonoff



Alan Newell



Herbert Simon



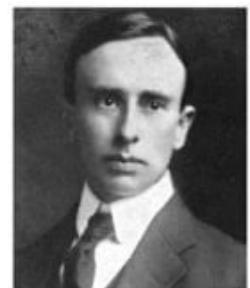
Arthur Samuel



Oliver Selfridge



Nathaniel Rochester



Trenchard More

A partir de esta conferencia, los siguientes años, se dieron sucesivos éxitos y avances (teniendo en cuenta que entonces se contaba con computadores y herramientas de computación bastante rudimentarios) fruto de investigaciones y multitud de proyectos con grandes expectativas. Los más importantes fueron:

- Creación del LISP en 1958 por John McCarthy (será el lenguaje de programación predominante en posteriores desarrollos de Inteligencia Artificial).
- Desarrollo de Micromundos (mundo de bloques) en 1959 por Minsky y Papert en el MIT.
- Construcción del demostrador de Teoremas de Geometría en 1959 por Herbert Gelernter.
- Investigación de los "sistemas expertos" en 1965 por Santford.
- Lanzamiento de ELIZA en 1966 (será el primer chatbot que implementa lenguaje natural).

La Década del Estancamiento (1970): A finales de la década de 1960 y principios de la década de 1970, la IA experimentó una etapa de estancamiento conocida como el "invierno de la IA". Los avances prometidos no se materializaron, y las expectativas sobre lo que la IA podría lograr superaron las capacidades tecnológicas y computacionales de la época. Esto llevó a una disminución en la financiación y el interés en el campo.

El Renacimiento de la IA (Décadas de 1980 y 1990): En la década de 1980, la IA experimentó un resurgimiento significativo debido a avances en la teoría y la computación. Se desarrollaron nuevas técnicas de razonamiento, representación del conocimiento y búsqueda heurística. Los sistemas expertos, que utilizan reglas y conocimiento específico de dominio para resolver problemas, se convirtieron en una aplicación exitosa de la IA en campos como la medicina y la ingeniería. Se produjo una rivalidad entre Estados Unidos y Japón para ver quién investigaba y desarrollaba más aplicaciones de IA. El principal gran hito en esta etapa fue la creación de R1, el primer sistema experto comercial, en 1982 gracias a McDermott (en 4 años de implantación supuso un ahorro de 40 millones de dólares al año).

Aprendizaje Automático y el auge de la IA moderna (finales de los 90 y década de 2000 en adelante): El siglo XXI ha sido testigo de una revolución en la IA, impulsada en gran parte por el auge del Aprendizaje Automático. Con la disponibilidad masiva de datos y avances en algoritmos, las máquinas ahora pueden aprender patrones complejos a partir de datos y mejorar su rendimiento a través de la experiencia. Esto ha llevado a avances significativos en campos como la visión por computadora, el procesamiento del lenguaje natural y el reconocimiento de voz.

La consagración definitiva de la Inteligencia Artificial llegó a finales de los 90. Con los siguientes grandes hitos, que veremos con algo más de detalle:

- El programa **Deep Blue** desarrollado por **IBM** logró vencer en **1997** al campeón del mundo en ajedrez, **Gari Kaspárov**.



Deep Blue fue una "supercomputadora" que desarrolló la empresa IBM en los años 90 del S.XX para jugar al ajedrez.

En febrero de 1996 se enfrentaron el entonces campeón del mundo, Gary Kaspárov contra la máquina. La primera partida la ganó Deep Blue, otras tres las ganó Kaspárov y dos más quedaron en tablas. Fue la primera máquina que derrotó a un experto jugador.

En realidad, siendo fieles al concepto de Inteligencia Artificial, esta máquina no se puede considerar exactamente como tal, pues "solo" era capaz de calcular a gran velocidad millones de posiciones posibles por segundo, pero le faltaba "intuición". Es decir, contaba con una tremenda base de datos (posibles jugadas, movimientos, etc), pero sus programadores no fueron jugadores de ajedrez expertos, por lo que la máquina no siempre elegía la jugada óptima.

¿Sabes cuántas jugadas posibles hay en el ajedrez? Son cerca de 10^{120} (diez elevado a ciento veinte... 1000000000000000000000000... así hasta 120 ceros).

Para hacerte una idea más real de la tremenda capacidad de cálculo de Deep Blue, se estima que la cantidad de átomos que existen en el Universo es de entre 10^{80} y 10^{82} .

El ajedrez es un juego complejo... ¿no crees?

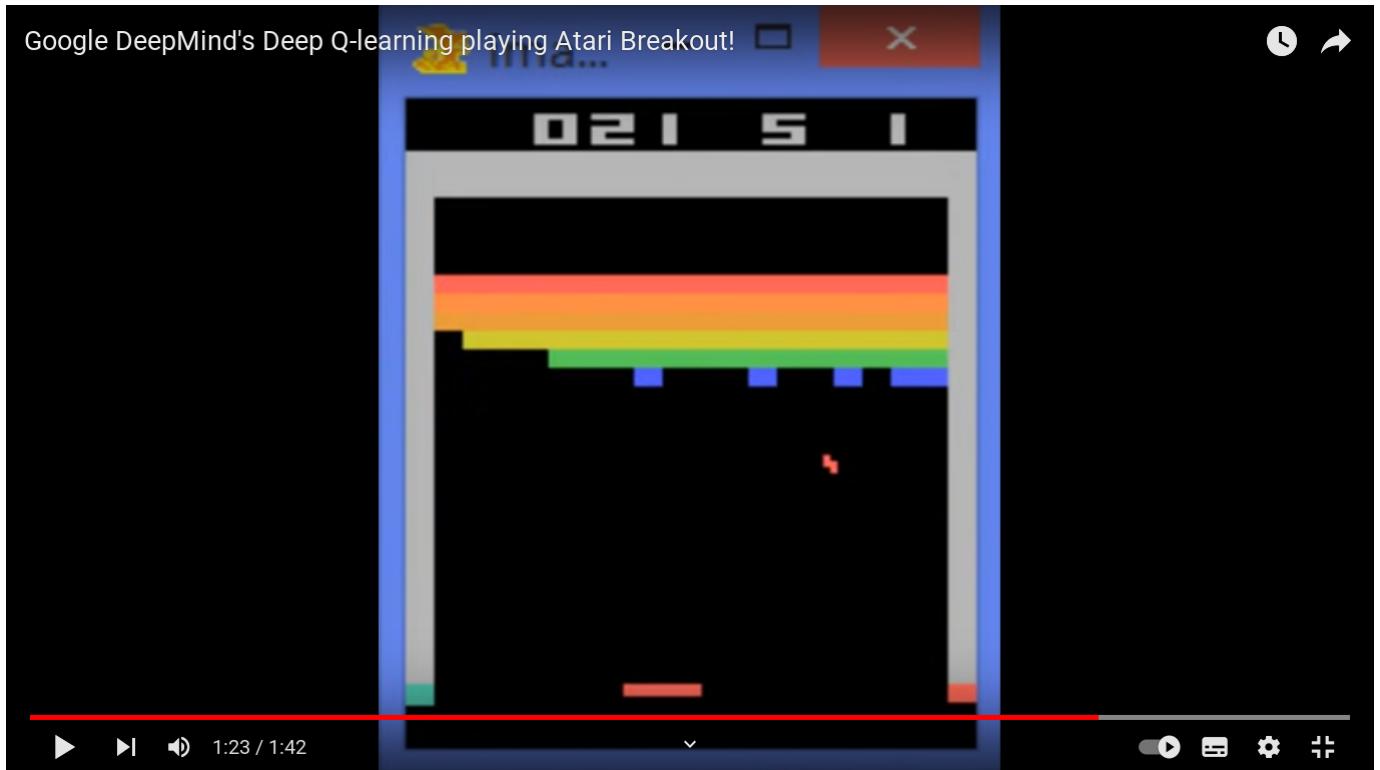
- El sistema **Watson**, también de **IBM**, logró ganar en **2011** el popular concurso televisivo **Jeopardy!** frente a los dos máximos campeones de este programa.



IBM continuó investigando en el campo de la Inteligencia Artificial (a la vez que sus ordenadores aumentaban su capacidad de cálculo) y, con lo aprendido con Deep Blue y otros desarrollos, creó Watson, una computadora que logró en 2011 ganar en Jeopardy!, uno de los concursos de conocimiento más famosos en Estados Unidos.

Watson era un sistema capaz de comprender y responder preguntas, en lenguaje natural (es decir, expresadas como habla cualquier humano, con variaciones y diferentes formas de expresar la misma idea). Contaba con una base de datos almacenada localmente (sin conexión a Internet) compuesta de enciclopedias, diccionarios, tesauros, artículos de noticias, obras literarias y otras bases de datos complementarias).

- La empresa **DeepMind** publicó "el video de los 500 millones de dólares". En dicho video mostraba cómo la red neuronal que había desarrollado aprende a jugar al **Arkanoid** de manera autónoma. Google acabó comprando esta empresa en **2014** por **500 millones** de dólares (de ahí el nombre del video).



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

DeepMind, una compañía inglesa creada en 2010, publicó a los pocos años de su creación el que se denominó "El vídeo de los 500 millones de dólares". En dicho vídeo (que puedes ver más arriba) mostraba cómo su Inteligencia Artificial había aprendido a jugar gracias a la técnica de entrenamiento autónomo (Machine Learning) por refuerzo al "Arkanoid" (un juego arcade del S.XX).

La red neuronal que desarrolló "aprendía" a jugar como un humano (con memoria a corto plazo, aplicando lo aprendido en cada partida a las siguientes). Así, en el vídeo podemos ver cómo en las primeras partidas descubre cómo debe mover para evitar perder. Mas adelante aprende a ganar puntos destruyendo ladrillos, y a "ganar" la partida alcanzando la máxima puntuación. Y finalmente "descubre" que si logra colar la pelota por un lateral hasta la parte superior de la pantalla gana en mucho menos tiempo.

Al poco tiempo de publicarse este vídeo recibió oferta de compra de Facebook, que no se materializó, y acto seguido de Google, que la compró por 500 millones de dólares (de ahí el nombre del vídeo).

Ya dentro de la matriz de Google continuaron profundizando en las técnicas de Aprendizaje Automático, logrando otro hito, como veremos más adelante.

- **Google liberó TensorFlow**, su librería para Machine Learning, en **2015**, permitiendo que cualquier persona pudiera acceder a sus servidores y crear su propio equipo con capacidad de autoprogramación y de aprender de forma autónoma.



https://www.youtube.com/watch?v=oZikw5k_2FM

En noviembre de 2015 Google liberó TensorFlow, el programa de Inteligencia Artificial que había desarrollado mejorando un sistema de aprendizaje automático anterior conocido como DistBelief.

Fue la primera vez que se ponía a disposición de cualquier usuario, investigador o empresa interesados en realizar sus propios experimentos de Inteligencia Artificial. Supuso un gran impulso tanto en el campo de la investigación (la propia comunidad de desarrolladores ayudó y sigue ayudando a mejorar y perfeccionar la herramienta) como en el de la democratización de la Inteligencia Artificial, haciéndola accesible para todos.

En la actualidad sigue utilizándose tanto para primeras aproximaciones al universo de la IA, como para desarrollar prototipos o ejercicios más complejos.

- La IA de **AlphaGo** de **Google** sorprendió a todos proponiendo en una partida de Go una jugada que nunca hubiera hecho un experto jugador humano... que en pocos movimientos más le dió la victoria.



Trailer: https://www.youtube.com/watch?v=8tq1C8spV_g

Documental completo: <https://www.youtube.com/watch?v=WXuK6gekU1Y>

Con subtítulos en español: <https://www.youtube.com/watch?v=GIJ7zr4sYx4>

Si jugar al ajedrez es complicado... el juego Go (de origen también oriental) lo es todavía más. La división de Google Deep Mind desarrolló una Inteligencia Artificial capaz de jugar a este juego. Pero la diferencia respecto a logros previos alcanzados por programas de IA capaces de jugar (al ajedrez, a un concurso de preguntas y respuestas...) es que lo hacía "con intuición".

En marzo de **2016** AlphaGo se enfrentó a **Lee Sedol**, que era uno de los mejores jugadores del momento. Ganó AlphaGo. Pero además lo hizo con una jugada (el movimiento 37 de esa partida) completamente inesperado y que ningún experto jugador humano hubiera hecho nunca. Es decir, que aunque la máquina estaba entrenada con registros de partidas reales jugadas por expertos, no siguió ninguna estrategia observada en su base de datos (contaba con 50 millones de partidas de Go entre humanos). Había aprendido a jugar "con intuición": aprendió de los ejemplos "humanos" pero descubrió formas de jugar nuevas (¡y eficaces!). La cara que se le puso a Lee Sedol al observar y analizar el movimiento de la máquina en ese turno 37 de la partida se hizo famosa.

- **Ian Goodfellow** presentó en **2014** su generador de imágenes basado en lo que conocemos como red **GAN**, logrando que un humano no sepa distinguir si se trata de imágenes reales o inventadas.

Las redes GAN (Generative Adversarial Networks) o generativas antagónicas presentadas por Ian Goodfellow en 2014 han permitido generar fotografías que parecen auténticas a cualquier observador humano.

Posteriormente este tipo de técnica (Aprendizaje Automático Supervisado, que veremos más adelante) también se ha aplicado a la generación de textos tal y como los escribiría un humano.

En esencia las redes GAN se componen de una red generadora (que crea la imagen, texto o diseño) y una red discriminadora (que determina si el resultado de la red generadora es aceptable o no). Ambas redes "compiten" entre ellas (la primera para "engañosamente" engañar a la segunda, y la segunda para detectar fallos en lo generado por la primera). El sistema se retroalimenta y perfecciona con cada iteración.

En esta web se muestra, cada vez que actualizas la página, la imagen de un rostro humano generado por IA: <https://thispersondoesnotexist.com/>. En algunos casos se notan cosas raras (en las pupilas, orejas, o fondos), pero en general suelen salir rostros que bien podrían corresponder con personas reales ¡Pero en realidad esas personas no existen!

- Desarrollo de **GPT3** por **OpenAI** a través de técnicas de Deep Learning.

GPT-3 es la tercera generación del Modelo de Predicción del Lenguaje que ha sido presentada en mayo de **2020**. Se trata de una Inteligencia Artificial "educada" para escribir cualquier tipo de texto, con cualquier tipo de estilo. A partir de unas pocas

palabras que le proporcionas explicando qué es lo que quieras te devuelve un texto complejo que trata sobre lo que le hayas pedido.

Lo más importante de esta tecnología son los **175 Billones de parámetros que utiliza** la para conseguir dar textos naturales (con aspecto de haber sido escritos por humanos).

- **GPT-4** representa la cuarta generación del Modelo de Predicción del Lenguaje, presentado en septiembre de **2022**. Esta versión ha experimentado avances significativos en la capacidad de generación de texto. Como una Inteligencia Artificial avanzada, GPT-4 ha sido entrenado para producir textos aún más naturales y coherentes, adaptándose a cualquier estilo y contenido requerido. Sorprendentemente, cuenta con una impresionante cantidad de **250 Billones de parámetros**, lo que le permite alcanzar un nivel de sofisticación sin precedentes en la creación de textos que parecen ser obra de escritores humanos expertos.

Aprendizaje Profundo (Deep Learning): El Aprendizaje Profundo, una rama del Aprendizaje Automático, ha sido uno de los desarrollos más destacados en la IA moderna. Las redes neuronales profundas, inspiradas en la organización del cerebro humano, han demostrado ser altamente efectivas en tareas como el reconocimiento de imágenes, la traducción automática y el juego de estrategia. La escalabilidad de los algoritmos de Aprendizaje Profundo, junto con la disponibilidad de potentes unidades de procesamiento gráfico (GPU), ha impulsado el rápido progreso en el campo.

IA en la Sociedad Actual: En la actualidad, la IA ha permeado en diversas áreas de nuestra vida cotidiana. Está presente en aplicaciones como motores de búsqueda en línea, asistentes virtuales, recomendaciones de productos y servicios, sistemas de navegación, automóviles autónomos y mucho más. La IA también está siendo utilizada en campos como la medicina para el diagnóstico y tratamiento, en finanzas para la detección de fraudes y en la industria para optimizar procesos de producción.

Desafíos Actuales y Futuros: Aunque la IA ha logrado avances impresionantes, todavía enfrenta desafíos significativos. Uno de ellos es la interpretabilidad y explicabilidad de los modelos de IA, especialmente en aplicaciones críticas donde las decisiones pueden tener un impacto importante en las vidas humanas. Otro desafío es el sesgo en los datos y la falta de diversidad, lo que puede llevar a resultados injustos o discriminatorios. A medida que la IA continúa avanzando, es esencial abordar estos desafíos y asegurar que su desarrollo y aplicación se realicen de manera ética y responsable.

¿La Inteligencia Artificial es buena o mala?

Piensa en diferentes momentos históricos en los que la humanidad ha desarrollado alguna tecnología: El dominio del fuego, la rueda, el hormigón, la pólvora, la imprenta, la radio, Internet...

La tecnología en sí misma no es ni buena ni mala. Son las personas que la conocen y controlan quienes pueden hacer un uso beneficioso o dañino de ellas.

¿Te suena la frase "Un gran poder exige una gran responsabilidad"? La IA nos da un poder tan grande (o mayor) que el Spiderman... Hemos de ser responsables al utilizarla.

El futuro de la IA

Los campos en los que más se ha desarrollado y aplicado la IA en estos últimos años son:

- Sistemas autónomos
- Aprendizaje Autónomo (Machine Learning)
- Aprendizaje Profundo (Deep Learning)
- Redes neuronales.
- Reconocimiento de patrones
- Procesado del lenguaje natural
- Desarrollo de chatbots
- Reconocimiento de emociones

En la actualidad se está trabajando (y se esperan mejoras en los próximos años) en campos como:

- Asistentes virtuales
- Traducción simultánea universal.
- Control de juegos con el pensamiento.

Y a medio plazo se prevé que la Inteligencia Artificial proporcione soluciones y mejoras en los siguientes ámbitos:

- Nueva generación de robots interconectados con la nube.
- Robots médicos autónomos.
- Asistentes personales robóticos.

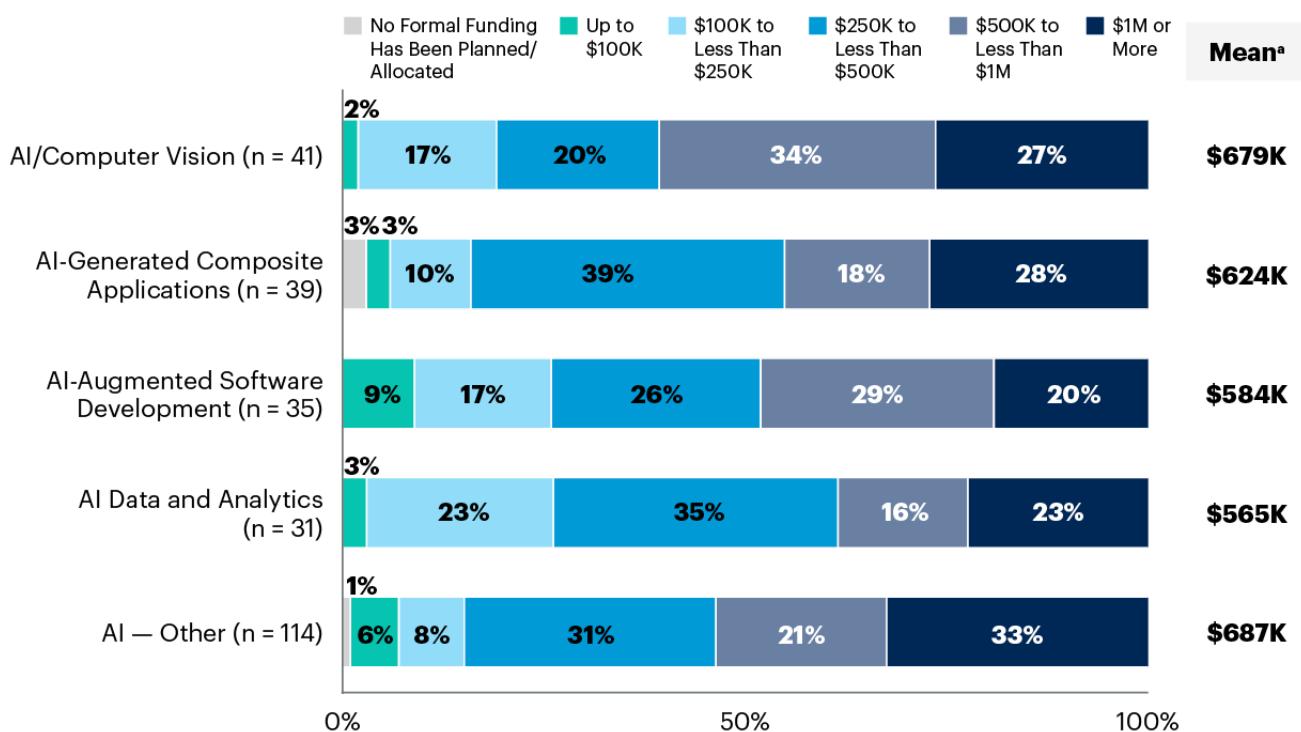
- Ciber-Seguridad cognitiva.

Y a largo plazo se vislumbra que puedan llegar a desarrollarse computadoras robóticas con forma y comportamiento humano.

Inversión en proyectos de IA

Como hemos visto la Inteligencia Artificial ya está demostrando de manera práctica los beneficios que puede proporcionar a empresas e instituciones. Las empresas tecnológicas fueron pioneras hace pocos años al incorporar en sus procesos y productos estas aplicaciones. Y en la medida en que todo el entramado empresarial y social se está digitalizando, la posibilidad de incorporar tecnologías inteligentes en cualquier sector está al alcance de casi cualquiera.

De hecho, en los últimos años la Inteligencia Artificial es el primer o segundo ámbito en el que más dinero están dispuestas a invertir las empresas (por delante de otras tecnologías emergentes como Internet de las Cosas, o los datos en la nube). En el siguiente gráfico puedes ver el nivel de financiación que se preveía dedicar en 2022 a incorporar y desarrollar Inteligencia Artificial en un grupo de empresas analizado por Gartner.



fuente: <https://www.gartner.com/en/newsroom/press-releases/2021-09-29-gartner-finds-33-percent-of-technology-providers-plan-to-invest-1-million-or-more-in-ai-within-two-years>

Limitaciones Prácticas Actuales

Aunque muchos de los argumentos filosóficos y científicos en contra de la inteligencia artificial pueden abordarse con respuestas lógicas, teóricas o basadas en la evidencia, es importante reconocer que existen limitaciones prácticas actuales en el campo de la IA que aún no se han superado por completo. Algunas de estas limitaciones incluyen:

- Complejidad de la mente humana:** La mente humana es increíblemente compleja, y aún no comprendemos completamente todos los aspectos de cómo funciona. La IA actual está lejos de replicar la complejidad y la plasticidad del cerebro humano.
- Memoria y recursos limitados:** Aunque las capacidades de almacenamiento y procesamiento de las computadoras han aumentado drásticamente, todavía estamos lejos de igualar la capacidad de almacenamiento y la eficiencia de procesamiento del cerebro humano.
- Falta de comprensión de la conciencia:** A pesar de los avances en neurociencia y cognición, aún no entendemos completamente la naturaleza de la conciencia y cómo emerge en el cerebro. Sin esta comprensión, es difícil replicarla en una máquina.
- Ética y responsabilidad:** La creación de inteligencia artificial plantea cuestiones éticas y de responsabilidad importantes, como el temor a la toma de decisiones no éticas o la falta de responsabilidad en caso de errores graves.
- IA en entornos no controlados:** La IA puede funcionar bien en entornos controlados y con datos bien estructurados, pero tiene dificultades para adaptarse a situaciones inesperadas o entornos no controlados.
- Creatividad e intuición:** Aunque se han logrado avances en la generación de contenido creativo por parte de las máquinas, la verdadera creatividad e intuición humana siguen siendo difíciles de replicar.

7. **Emociones y empatía:** La comprensión y expresión emocional, así como la empatía, son aspectos desafiantes de la inteligencia humana que no se han logrado de manera completa en la IA.
8. **Interacción social humana:** La comprensión del lenguaje natural, la interacción social y la percepción de matices emocionales en el contexto de la comunicación humana son desafíos actuales para la IA.

Es importante tener en cuenta estas limitaciones y reconocer que la IA actual está lejos de igualar la inteligencia humana en todos sus aspectos. Sin embargo, esto no implica que no haya avances significativos en el campo de la IA, ni que no se puedan superar algunas de estas limitaciones en el futuro.

Principios de Sistemas Inteligentes

Así pues, puede entenderse por sistema informático el conjunto de cosas (hardware y software) y al conjunto de reglas (procedimientos) que de manera conjunta se emplean para el fin último de adquirir, almacenar, procesar y representar la información de manera automatizada.

El **hardware** incluye computadoras o cualquier tipo de dispositivo electrónico, principalmente constituido alrededor de semiconductores como memoria, procesadores, sistemas de almacenamiento externo, etc.

El **software** incluye al sistema operativo, firmware y aplicaciones. También se puede considerar parte del sistema a quien hace uso del hardware: los programadores y los usuarios.

La potencia y eficacia de un sistema de la información radica en la correcta correlación de una gran cantidad de datos ingresados mediante procesos específicos para cada campo o tarea, con el objetivo de producir información para la posterior toma de decisiones. Un sistema informático se destaca por su diseño, facilidad de uso, flexibilidad, mantenimiento automático de los registros, apoyo en la toma de decisiones críticas y la conservación del anonimato en informaciones irrelevantes.

Por todo ello, si la inteligencia artificial es un subconjunto de la informática (en el sentido de computación o ciencia de las tecnologías de la información), un sistema de inteligencia artificial ha de ser por extensión un subconjunto dentro de los sistemas informáticos.

Se denominará, por tanto, sistema inteligente a un programa o conjunto de programas de computación que reúne características y comportamientos asimilables al de la inteligencia humana o animal.

Para que un sistema informático pueda ser considerado un sistema inteligente, habrá de tener las características que se enumeran a continuación:

1. En primer lugar y como clara obviedad el sistema debe poseer inteligencia, entendiendo como ello el **ser un sistema inteligente**.
2. El hecho de ser un sistema implica que ha de disponer de sistematización entre sus componentes, es decir, **las partes del sistema han de tener correlaciones con otros elementos del mismo sistema**.
3. El sistema ha de ser capaz de **cumplir uno o varios objetivos**, o sea, una cierta situación, condición o estado que el sistema inteligente busca lograr.
4. El sistema ha de disponer de **capacidad sensorial**, para ello ha de tener una parte que sea capaz de recibir comunicaciones del entorno: el sistema inteligente ha de poder reaccionar ante el entorno y sus variaciones, lo que se consigue normalmente mediante sensórica.
5. El sistema ha de exhibir capacidad de conceptualización (entendiendo como concepto al elemento básico del pensamiento), lo que implica la **necesidad de poder almacenar información**. Para poder conceptualizar es necesario el desarrollo de niveles de abstracción.
6. El sistema ha de disponer de **procedimientos y métodos con reglas de actuación**, por tanto, el sistema ha de ser capaz de relacionar situaciones y consecuencias de acciones.
7. El aprendizaje es la capacidad más importante y exclusiva de un sistema inteligente. **El sistema aprende nuevos conceptos a partir de la información recibida de los sentidos, las reglas conocidas y la experiencia**. El aprendizaje también es la capacidad de detectar relaciones (patrones) entre la «situación» y la parte «situación futura» de una regla de actuación.

Los primeros sistemas inteligentes, como los sistemas de expertos, no cumplen todas estas características por lo que reciben el nombre de sistemas de inteligencia incompletos.

La inteligencia artificial (IA) es un área de cambio social, que transforma rápidamente los hábitos y costumbres de las sociedades, y por ello ha de prestarse mucha atención a sus posibilidades, y marcar una serie de límites éticos. Los principios fundamentales o empleos éticos de la IA son los siguientes:

1. **La IA debe estar libre de prejuicios**, tanto en el caso inferencia como en el entrenamiento. Entrenar datos de manera equivocada puede generar discriminantes negativas que alejan la toma de decisiones de la realidad. Además, en conjunto, toda IA debe programarse de manera que no se usen conjuntos sesgados, y evitar discriminaciones algorítmicas por medio de métricas

imparciales avaladas por expertos humanos. Aunque parece algo lejano, se introducen continuamente sesgos en el aprendizaje de las IA, muchas veces de manera involuntaria e inadvertida.

Sistemas de ayuda médica entrenados en varones blancos de entre 30 y 50 años de edad podrán dar resultados buenos sobre el grupo para el que se ha desarrollado, pero pueden actuar no tan correctamente en otras categorías.

- 1. Ayudar a ayudar.** Se debe de identificar de forma clara la responsabilidad de las decisiones tomadas por los sistemas autónomos. Los usuarios de las IA, especialmente en caso de grandes corporaciones y gobiernos, han de aprender a estimar y evaluar las consecuencias positivas y negativas de la implantación de sistemas de inteligencia artificial, en la sociedad en su conjunto, dado el gran poder de cambio que generan.
- 2. Uso de algoritmos abiertos.** Para poder confiar en la respuesta de una IA es preciso tener acceso limpio al algoritmo de entrenamiento y de toma de decisiones que posee, lo que comporta acceso a todo su modelo matemático. Supone la única manera de poder explicar el funcionamiento que tendrá la misma.
- 3. Seguridad, privacidad y confiabilidad.** Dado que las IA hacen uso de gran cantidad de datos, se ha de velar por la transparencia y la privacidad en el uso de los mismos.

Un asistente virtual ha de garantizar que las conversaciones escuchadas no se filtrarán ni difundirán a terceros.

- 1. Bien común.** Ningún sistema de IA debería ser desplegado si al hacerlo se atenta contra el bien común.

Los agentes inteligentes son entidades capaces de percibir su entorno a través de sensores y actuar en él mediante efectores para alcanzar objetivos específicos. Un agente puede ser tan simple como un programa que juega ajedrez o tan complejo como un vehículo autónomo que navega por las calles de una ciudad. Los agentes inteligentes se basan en la idea de que una "inteligencia" puede emerger de la interacción entre un sistema y su entorno, sin necesidad de una planificación o conocimiento exhaustivo previo.

Componentes de un Agente Inteligente:

- 1. Sensores (S):** Los sensores son dispositivos que permiten al agente percibir información sobre su entorno. Pueden incluir cámaras, micrófonos, sensores de temperatura, GPS, entre otros. La información que los sensores recopilan se utiliza para representar el estado actual del entorno.
- 2. Actuadores (A):** Los actuadores son los medios mediante los cuales el agente interactúa con su entorno. Pueden ser ruedas en un robot, motores en un brazo robótico o simplemente salidas de datos en un sistema de software. Los actuadores permiten que el agente tome decisiones y realice acciones para alcanzar sus objetivos.
- 3. Función del Agente (f):** La función del agente representa el comportamiento del agente en función de las percepciones que recibe. Toma como entrada el estado actual del entorno y devuelve una acción que el agente debe ejecutar. Esta función puede ser simple o compleja, dependiendo de la complejidad de la tarea que el agente debe realizar.
- 4. Arquitectura (A):** La arquitectura del agente se refiere a cómo se organiza el agente en términos de sus componentes y cómo interactúan entre sí. Puede haber diferentes arquitecturas según la complejidad de la tarea y los requisitos de rendimiento.

Agente Inteligente:

Un ejemplo sencillo de un agente inteligente es un sistema de navegación GPS en un automóvil. En este caso:

- **Sensores (S):** El sistema de navegación utiliza sensores GPS para recibir información sobre la ubicación actual del automóvil y sensores de velocidad para conocer su velocidad y dirección.
- **Actuadores (A):** Los actuadores son los mecanismos que permiten al sistema de navegación proporcionar instrucciones al conductor para alcanzar el destino deseado, como la pantalla de navegación o las indicaciones de voz.
- **Función del Agente (f):** La función del agente en este caso podría ser bastante simple: recibir la ubicación actual y el destino deseado, calcular la ruta más rápida y segura y guiar al conductor a lo largo del camino.
- **Arquitectura (A):** La arquitectura del sistema de navegación podría ser una combinación de algoritmos de planificación de rutas, sistemas de reconocimiento de voz para recibir comandos del conductor y sistemas de visualización para mostrar las indicaciones.

A continuación, un diagrama para ilustrar la interacción de un agente inteligente con su entorno:

```
flowchart LR
    S((Sensores)) --> f((Función<br>del Agente))
    f --> A((Actuadores))
    A --> E((Entorno))
    E --> S
```

En el diagrama, los sensores recopilan información del entorno, que se utiliza como entrada para la función del agente. La función del agente procesa la información y toma una decisión sobre qué acción ejecutar. Luego, los actuadores implementan la acción en el entorno, lo que puede cambiar su estado. El ciclo se repite continuamente, permitiendo al agente inteligente interactuar y adaptarse a su entorno para lograr sus objetivos.

Tipos de Inteligencia Artificial. Escuelas y clasificaciones

En la actualidad la evolución de la Inteligencia Artificial comprende un campo tan amplio, con tantas ramas, que es complicado poder atender a una única clasificación. De hecho encontramos diferentes clasificaciones según la diferente visión con la que se aborda dicha tecnología. Las hay más filosóficas, más técnicas, y según su aplicación.

Dentro de las distintas clasificaciones que vamos a ver, hay algunas tipologías que están definidas "en teoría" aunque aún no exista ninguna aplicación IA de esa clase.

Según tareas a resolver

La primera clasificación de la Inteligencia Artificial que vamos a ver se basa en qué tipo de tareas nos ayuda a resolver o ejecutar.

Cuando hablamos de tarea a resolver nos referimos, por ejemplo, a si pretendemos que la IA sea capaz de jugar al ajedrez en un ordenador o si pretendemos que sea capaz de gestionar una cocina sin intervención humana (desde el abastecimiento de alimentos, procesado, decisión de qué cocinar en cada momento, limpieza y mantenimiento de utensilios, resolver imprevistos o accidentes...).

Hay tareas sencillas, concretas, y puntuales que son relativamente sencillas de programar, mientras que hay tareas complejas en las que, además influyen muchos factores exteriores (sentimientos, contexto, moral, ética, creencia religiosa...).

Seguro que eres capaz de intuyendo que nuestros programas de Inteligencia Artificial actuales son más bien de los que resuelven tareas en un entorno muy concreto y acotado, mientras que hay que irse a las películas o series de ficción para poder hablar de algún caso de Inteligencia Artificial capaz de actuar en escenarios complejos, cambiantes y con "conciencia".

Esta clasificación según tareas es una aproximación bastante simple, con dos opciones:



Como veremos en la explicación más detallada de cada una de estas dos posibles categorías de Inteligencia Artificial, en la actualidad aún no hemos sido capaces de desarrollar ninguna IA Fuerte. Todo lo que conocemos en este momento quedaría dentro de la clasificación de IA Débil.

INTELIGENCIA ARTIFICIAL DÉBIL (O ESTRECHA)

IA Débil también conocida como **IA estrecha**, se define como la inteligencia artificial racional que se centra típicamente en una tarea estrecha. Es decir orientada a resolver problemas muy concretos, en un entorno perfectamente acotado.

Por tanto consideramos que este tipo de Inteligencia Artificial débil es limitada, pues no es capaz de adaptarse o asumir cambios respecto a lo que se le ha programado.

Los asistentes virtuales (Siri, Ok Google, Alexa, etc.) son un ejemplo bastante ilustrativo de hasta dónde es capaz de llegar la Inteligencia Artificial débil. Cualquiera de ellos opera dentro de un rango de respuestas limitado, definido en su base de datos. En realidad "la máquina" no tiene inteligencia genuina. No es capaz de aprender, ni de tener en cuenta el entorno o el contexto en el que se le realizan las preguntas. No tiene conciencia, ni mucho menos vida propia. Sin duda es un tipo bastante sofisticado de IA débil, pero llega un punto en el que no es capaz de responder cierta clase de preguntas, y, salvo que cambiáramos su base de datos y programación nunca sería capaz de llegar a encontrar por sí misma respuestas adecuadas a dichas preguntas.

De hecho, uno de los principales entretenimientos más habituales cuando nos ponemos "a charlar" con un asistente virtual es intentar llevarla al límite... A ver en qué tipo de pregunta, cada cual más compleja o absurda, es incapaz de responder. O, sin llegar al límite de no encontrar respuesta, puede llegar a dar respuestas molestas o inadecuadas.

Desde el punto de vista de esta clasificación (por tarea a resolver), **las características** de la Inteligencia Artificial débil son:

- **Ya existen en la vida real:** Como hemos comentado, los asistentes virtuales, programas como Watson o Alpha Go que vimos en la unidad anterior.
- Se orientan a **resolver problemas muy concretos:** El programa que "sabe" jugar al Go, no sabe hacer otra cosa. Ni tiene posibilidades de aprender a jugar a otra cosa, por muy similar que sea.
- Son **reactivas:** No tienen iniciativa, es necesario que se desencadene la acción que tienen programada para que se inicie su rutina. En el ejemplo del asistente virtual, tiene programado responder cuando le preguntas, y por tanto nunca tomará la iniciativa de ofrecerte nada sin que tú lo actives previamente.
- **No son flexibles:** Colapsan si se encuentran en un caso no previsto en su programación.
- Quedan **limitadas por lo que programa un humano:** Es el humano quien programa lo que "tiene que pensar" la máquina. Si el humano no programa deja sin considerar ciertas opciones o posibles situaciones, la IA nunca será capaz de suplirlo o aprenderlo sobre la marcha por sí misma.
- Se **programan con pocas redes neuronales:** Hablaremos más adelante sobre las redes neuronales. Por el momento es suficiente entender que el nivel de computación compleja que requieren este tipo de Inteligencias Artificiales es menor que otros casos.
- **No razonan, solo computan:** No tienen en cuenta ningún factor moral, contextual, circunstancial, emocional... que a un humano le haría reaccionar de manera diferente.
- La máquina está programada para alcanzar tal objetivo o funcionar de tal manera, y así lo hará sin "entender" lo que está haciendo. Por tanto: **no tiene conciencia**.
- **Aprenden a base de ejemplos:** Necesita conocer muchos ejemplos de lo que tiene que hacer (la base de datos), con todas las variantes posibles. Por ejemplo, en la máquina que juega al Go, se la "entrenó" con 50 millones de partidas de dicho juego.
- **Son repetitivas:** No se cansan nunca, son implacables, siempre la misma rutina. No salen de su marco de trabajo: Y esto supone que pueden ocuparse de tareas mecánicas, repetitivas, "aburridas" para sustituir al humano mejorando rendimiento y precisión. Pero necesitará siempre una supervisión humana que vaya decidiendo cómo adaptar el programa a las cambiantes circunstancias.

Según se mire la Inteligencia Artificial débil podría llegar a ser peligrosa. Porque este tipo de IA, al no tomar en consideración todo un contexto amplio, ni seguir las reglas sociales, éticas,... ejecuta las tareas para las que se le ha entrenado con eficacia y contundencia. No evalúa las consecuencias como lo hacemos los humanos, considerando un espectro amplio de efectos y relaciones. Por eso, es una opción incompleta, inestable y peligrosa si no se utiliza con prudencia, o si quien la programa pretende, precisamente, causar mal.

INTELIGENCIA ARTIFICIAL FUERTE

La **Inteligencia Artificial fuerte (IAF)** o general o (IAG), desde el punto de vista de la tarea a resolver, sería aquella que iguala o excede la inteligencia humana promedio. Sería capaz de realizar con éxito cualquier tarea intelectual del ser humano, teniendo en cuenta todos los factores y matices que pueden intervenir cuando una persona toma decisiones en cada momento mientras realiza una tarea.

En comparación con la Inteligencia Artificial débil, las características de la fuerte son:

- **No existe en la realidad:** Si quieres "ver" cómo sería puedes recurrir a personajes de ficción como T-800, Wall-E o J.A.R.V.I.S. Resolverán problemas abiertos: Deberían poder abarcar múltiples posibles tareas, distintas unas de otras (reparar una puerta, ir a recoger a los niños del colegio, regar las plantas, darte conversación...).
- **Serán proactivas:** En función de la misión u objetivo que tenga, y de las circunstancias, iniciará cualquier tipo de rutina sin esperar a que un humano se lo pida o esté pendiente.
- **Serán flexibles:** Podrán encontrar similitudes entre algo que conocen y algo que se le parezca un poco. Por ejemplo, aunque inicialmente solo haya sido programada para saber andar, será capaz de aprender a correr sin necesidad de intervenir en su programa.
- **Se autoprogramarán:** Serán capaces de detectar sus propios límites y se regularán a sí mismas para no excederlos.
- **Usarán muchas redes neuronales:** Y además podrán entrar en conflicto entre ellas en algunas ocasiones. Esto quiere decir que necesitarán una capacidad de almacenaje de información y cómputo que aún hoy no hemos llegado a alcanzar.
- **Imitarán el comportamiento humano:** Serán capaces de razonar, y por tanto, de alcanzar algún tipo de conciencia.
- **Aprenderán como las personas:** Podrán recordar datos, observar nuevas situaciones y encontrar relaciones entre diferentes acciones. Esto quiere decir que si saben jugar al ajedrez y "observan" el juego de las damas, podrán aprender a jugar a las damas basándose en lo que saben sobre jugar al ajedrez.
- **Serán capaces de aprender nuevas tareas:** Modificarán la tarea o cómo realizan la tarea para adaptarse a las circunstancias.
- **Serán capaces de adaptarse a nuevos escenarios:** Podrán adaptarse a cambios y nuevas situaciones para seguir cumpliendo su objetivo.

Este tipo de IA es la que sería capaz de analizar cualquier situación y deducir el conjunto de acciones más adecuado para dicha situación y contexto. Lo mismo sabría conducir un coche, que resolver una ecuación matemática o mantener una conversación sobre un tema concreto.

Aunque aún no existe este tipo de IA, todas las empresas e instituciones dedicadas a la investigación y desarrollo de IA están buscando formas de avanzar hacia este tipo de Inteligencia Artificial. De momento, al menos, se está trabajando en conseguir una IAF en el campo de los asistentes virtuales.

Sin duda uno de los ámbitos más ambiciosos de aplicar esta IAF (los asistentes virtuales humanoides) necesitan contar también con otras ramas científicas como son la robótica y mecatrónica.

Escuelas de Pensamiento.

En el ámbito de la Inteligencia Artificial más moderna podemos encontrar dos escuelas de pensamiento:

- Inteligencia Artificial **Convencional**.
- Inteligencia Artificial **Computacional**.

Estas dos escuelas difieren en la ciencia que hay tras los procesos que siguen para llegar a los resultados esperados. Pero, con los avances que se están dando en los recursos que utiliza la segunda, muchas de las aplicaciones que tenía la primera, están siendo llevadas al campo computacional.

INTELIGENCIA ARTIFICIAL CONVENCIONAL

Se conoce también como Inteligencia Artificial **simbólico-deductiva**. Está basada en el análisis formal y estadístico del comportamiento humano ante diferentes problemas:

- **Razonamiento basado en casos:** Ayuda a tomar decisiones mientras se resuelven ciertos problemas concretos y, aparte de que son muy importantes, requieren de un buen funcionamiento.
- **Sistemas expertos:** Infieren una solución a través del conocimiento previo del contexto en que se aplica y ocupa de ciertas reglas o relaciones.
- **Redes bayesianas:** Propone soluciones mediante inferencia probabilística.
- **Inteligencia artificial basada en comportamientos:** Esta inteligencia contiene autonomía y puede auto-regularse y controlarse para mejorar.
- **Smart process management:** Facilita la toma de decisiones complejas, proponiendo una solución a un determinado problema al igual que lo haría un especialista en dicha actividad.

Esta rama de la Inteligencia Artificial ha sido la que ha proporcionado la mayoría de algoritmos que conocemos como "automatización", y básicamente se sirven de sistemas con reglas condicionales y estadística avanzada.

INTELIGENCIA ARTIFICIAL COMPUTACIONAL

La Inteligencia Computacional (también conocida como IA **subsimbólica-inductiva**) implica desarrollo o aprendizaje interactivo (por ejemplo, modificaciones interactivas de los parámetros en sistemas de conexiones). El aprendizaje se realiza basándose en datos empíricos, utilizando métodos computacionales inspirados en procesos de la naturaleza, que permiten alcanzar soluciones aptas a problemas complejos que los modelos tradicionales no pueden resolver por no existir una solución analítica, por no contar con todos los parámetros necesarios o porque el problema es en sí estocástico y precisa de una aproximación envolvente en vez de convergente.

Esta corriente ha sido la que impulsó hace pocos años lo que conocemos como "Aprendizaje Automático" o "Machine Learning", que es la técnica que más se está utilizando actualmente en desarrollos de IA.

Algunas técnicas de esta escuela son:

- **Máquina de vectores soporte:** sistemas que permiten reconocimiento de patrones genéricos de gran potencia.
- **Redes neuronales:** sistemas basados en redes de unidades de computación lineal para simular computación no lineal
- **Modelos ocultos de Markov:** aprendizaje basado en dependencia temporal de eventos probabilísticos.
- **Sistemas difusos:** técnicas para lograr el razonamiento bajo incertidumbre
- **Computación evolutiva:** también conocidos como **algoritmos genéticos**, aplica conceptos inspirados en la biología, tales como población, mutación y supervivencia del más apto para generar soluciones sucesivamente mejores para un problema.

Clasificación Stuart J. Russell y Peter Norvig

Stuart J. Russell y Peter Norvig, investigadores informáticos, publicaron en 1995 su libro "**Artificial Intelligence: A Modern Approach**", que se ha convertido en el libro de texto fundamental en cientos de universidades a nivel mundial (ya lleva varias

ediciones publicadas). Plantean cuatro categorías básicas partiendo de un enfoque de génesis del acto inteligente, del origen y proceso por el cual se llega al comportamiento inteligente.

Sistemas Cognitivos: Piensan como humanos, intentan emular el proceso humano → Proceso de toma de decisiones, resolución de problemas, y el propio paradigma del aprendizaje.

Test de Turing: Actúan como humanos, intentan emular el comportamiento humano (sin pasar por el pensamiento o razonamiento que conduce a dicho comportamiento) → A nivel práctico se aplica en la robótica y sistemas de actuadores en el mundo físico.

Leyes del pensamiento: Piensan con razonamientos. Cumplimiento exacto de las leyes del razonamiento lógico, teniendo en cuenta todos los factores que afectan a la cuestión. No hemos llegado a esto aún. Sería el caso de los sistemas expertos. Solo son posibles aproximaciones para campos de investigación muy especializados y acotados.

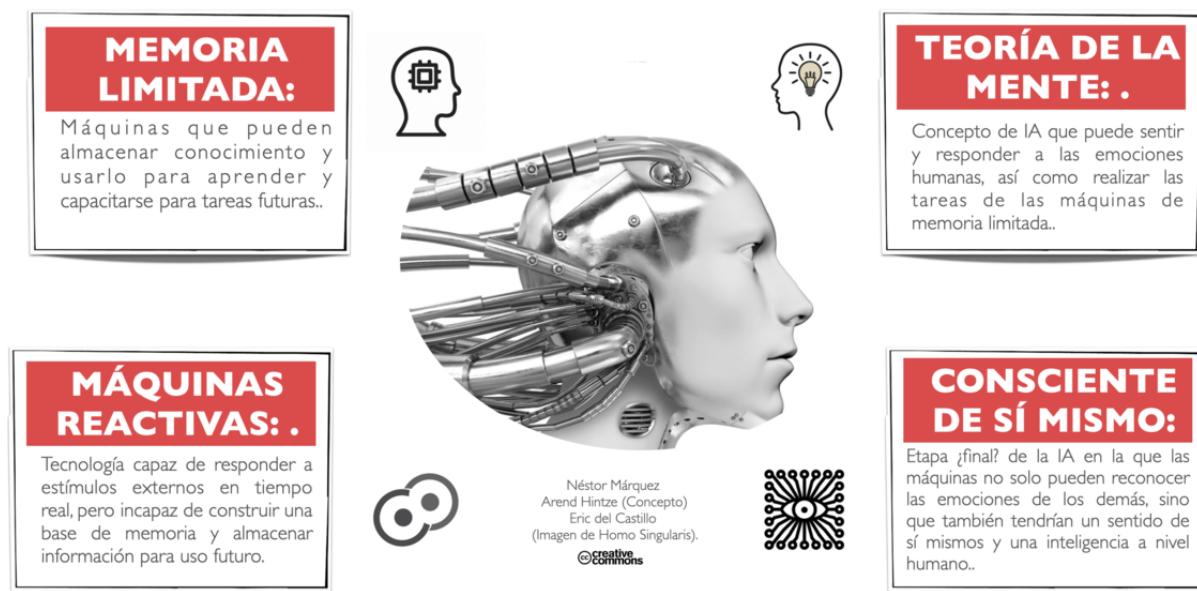
Agentes inteligentes: Actúan racionalmente (sin pasar por el proceso de razonamiento lógico).

Los dos últimos requieren una capacidad de cómputo muy importante, a veces, aún, inaccesible.

Clasificación Hintze

En noviembre de **2016**, **Arend Hintze**, profesor de la Universidad de Michigan e investigador en el campo de la Inteligencia Artificial, escribió un artículo titulado: "**Understanding the four types of AI, from reactive robots to self-aware beings**" (Comprendiendo los cuatro tipos de IA, desde los robots reactivos a los seres auto-conscientes), en el que sintetizaba toda la evolución de los últimos desarrollos y avances en materia de Inteligencia Artificial para aportar una clasificación más realista y concreta para los tipos de entidades que existen o que se aspira a crear.

4 TIPOS DE INTELIGENCIA (ARTIFICIAL)



MÁQUINAS REACTIVAS

Los tipos más básicos de sistemas de IA son puramente reactivos. No tienen la capacidad de formar recuerdos. Tampoco pueden utilizar experiencias pasadas en las que basar las decisiones actuales.

Deep Blue (descrita en el tema anterior) fue una supercomputadora creada por IBM. Fue capaz de vencer al ajedrez al gran maestro internacional Garry Kasparov. Ocurrió a fines de la década de 1990 y es el ejemplo perfecto de este tipo de máquina.

Puede identificar las piezas en un tablero de ajedrez y saber cómo se mueve cada una.

Puede realizar predicciones sobre los mejores movimientos y elegir el mejor de todas las posibilidades.

Pero no tiene ningún concepto del pasado. Tampoco posee recuerdos de lo que ha sucedido antes. Aparte de una regla de ajedrez, Deep Blue ignora todo antes del momento presente. Todo lo que hace es enfocar las piezas del tablero en tiempo real y elegir entre los siguientes movimientos posibles.

En el caso de que se trate de una Inteligencia Artificial reactiva aplicada a una máquina capaz de "conversar" es importante que el usuario sepa que está tratando con una máquina, pues de lo contrario se suelen crear falsas expectativas sobre lo que puede esperar de dicha conversación.

MEMORIA LIMITADA

El segundo tipo de Inteligencia Artificial que contempla la clasificación de Hintze se caracteriza por que sí maneja máquinas que pueden mirar hacia el pasado. Los vehículos autónomos ya hacen algo parecido. Por ejemplo, observan la velocidad y dirección de otros autos, y en base a la memorización de dicha información toman decisiones en el futuro inmediato.

Digamos que estas observaciones se agregan a las representaciones preprogramadas para la memoria de estos coches. Se incluyen marcas de carril, semáforos y otros elementos importantes, como curvas en la carretera.

También se añaden experiencias como cuando el automóvil decide en qué momento cambiar de carril para evitar interrumpir a otro conductor o ser embestido por un automóvil cercano.

Pero estas simples piezas de información sobre el pasado son solo transitorias. No se guardan como parte de la biblioteca de experiencias del automóvil. En estos tipos de inteligencia artificial, la máquina no puede compilar la experiencia durante años, como lo hace un humano.

TEORÍA DE LA MENTE

¿Podemos construir sistemas de Inteligencia Artificial que construyan representaciones completas, recordar sus experiencias y aprender cómo manejar situaciones nuevas?

Llegamos a un punto en el que nos acercamos más a los tipos de Inteligencia Artificial que deseamos en un futuro. Las máquinas de esta clase son más avanzadas. No solo forman representaciones sobre el mundo, también sobre otros agentes o entidades.

En psicología, esto se denomina 'teoría de la mente'. Implica la comprensión de que las personas, las criaturas y los objetos en el mundo pueden tener pensamientos y emociones que afectan a su propio comportamiento. Esto es crucial para la forma en que los humanos formamos sociedades, porque nos permite la interacción social.

Si las máquinas van a andar entre nosotros, deberán tener una comprensión sobre cómo pensamos y cómo sentimos. Además deberán llegar a saber qué esperamos y cómo queremos que nos traten. Tendrán que ajustar su comportamiento en consecuencia.

Como habrás podido intuir, este tipo de máquinas aún no existen. Igual que la IA Fuerte es aún algo que se sabe cómo funcionará pero que no hemos llegado a desarrollar todavía.

AUTOCONCIENCIA

El paso final del desarrollo de la IA es construir sistemas que puedan formar representaciones sobre sí mismos. En última instancia, los investigadores de la Inteligencia Artificial tendrán que comprender no solo la conciencia, sino también construir máquinas que la tengan.

Los seres conscientes son conscientes de sí mismos, conocen sus estados internos y pueden predecir los sentimientos de los demás. Es probable que estemos lejos de crear máquinas que sean conscientes de sí mismas. Sin embargo, los esfuerzos se enfocan hacia la comprensión de la memoria, el aprendizaje y la capacidad de basar las decisiones en experiencias pasadas.

Este es un paso importante para entender la inteligencia humana por sí misma. Es crucial para diseñar o desarrollar máquinas que sean más excepcionales para clasificar lo que ven frente a ellas.

Los cuatro tipos de inteligencia artificial dan una idea sobre las intenciones que el hombre tiene acerca del futuro de la máquina. Puede que estemos muy lejos de la Inteligencia Artificial autoconsciente. No obstante, está claro que eso es lo que se persigue en última instancia.

Utilización de modelos de Inteligencia Artificial

En esta sección, se explorarán los requisitos básicos de un sistema de resolución de problemas y los diferentes modelos de sistemas de Inteligencia Artificial, incluyendo la automatización de tareas, sistemas de razonamiento impreciso y sistemas basados en reglas.

Requisitos básicos de un sistema de resolución de problemas

Los sistemas de resolución de problemas basados en Inteligencia Artificial deben cumplir con ciertos requisitos fundamentales para ser efectivos y proporcionar soluciones precisas y útiles:

1. **Representación del Problema:** La representación adecuada del problema es crucial para la resolución exitosa. Los sistemas de IA deben seleccionar una estructura de datos y un modelo que reflejen de manera fiel el dominio del problema. Por ejemplo, para el procesamiento del lenguaje natural, se pueden utilizar modelos basados en redes neuronales que convierten el texto en representaciones vectoriales.
2. **Razonamiento y Toma de Decisiones:** Los sistemas de IA deben ser capaces de razonar sobre la información disponible y tomar decisiones informadas para llegar a una solución. Esto implica aplicar técnicas lógicas, de aprendizaje automático o de búsqueda heurística, según la naturaleza del problema.
3. **Aprendizaje y Adaptabilidad:** La capacidad de aprender de la experiencia y adaptarse es esencial para mejorar el rendimiento de un sistema de IA con el tiempo. El aprendizaje automático y el aprendizaje por refuerzo son enfoques comunes utilizados para habilitar esta funcionalidad.
4. **Eficiencia Computacional:** Los sistemas de resolución de problemas deben ser eficientes en términos computacionales para proporcionar respuestas rápidas y escalables a problemas complejos. Esto implica el uso óptimo de algoritmos y técnicas de optimización para reducir el tiempo de ejecución y los recursos necesarios.
5. **Interacción con Usuarios:** Los sistemas de Inteligencia Artificial deben permitir la interacción con los usuarios de una manera comprensible y natural. Esto implica el desarrollo de interfaces de usuario amigables que faciliten la comunicación y la comprensión mutua entre humanos y sistemas de IA.

Modelos de sistemas de Inteligencia Artificial

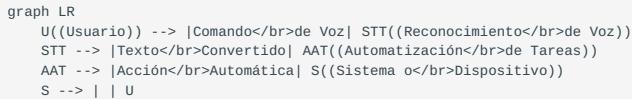
AUTOMATIZACIÓN DE TAREAS

La automatización de tareas es uno de los enfoques más comunes de la Inteligencia Artificial. En este modelo, se desarrollan sistemas que pueden realizar tareas específicas sin intervención humana directa. Algunos ejemplos de automatización de tareas son:

- **Reconocimiento de Voz:** Los sistemas de reconocimiento de voz convierten el habla en texto y pueden automatizar la transcripción de documentos o comandos de voz en dispositivos. Por ejemplo, aplicaciones de reconocimiento de voz como Google Speech-to-Text o Microsoft Azure Speech Service permiten convertir grabaciones de voz en texto escrito de manera automatizada.
- **Detección de Fraude:** Algoritmos de aprendizaje automático pueden analizar patrones de datos financieros y detectar transacciones sospechosas o fraudulentas. Por ejemplo, instituciones financieras utilizan modelos de aprendizaje automático para identificar patrones de comportamiento inusuales que podrían indicar actividades fraudulentas.

La automatización de tareas no solo mejora la eficiencia en diversas industrias, sino que también reduce la carga de trabajo manual y permite a los humanos centrarse en tareas más creativas y estratégicas.

A continuación, un diagrama para ilustrar el proceso de automatización de tareas mediante un sistema de reconocimiento de voz:



SISTEMAS DE RAZONAMIENTO IMPRECISO

Los sistemas de razonamiento impreciso permiten el manejo de incertidumbre y vaguedad en los datos y la toma de decisiones. Estos sistemas son útiles cuando los datos son incompletos o inciertos y se basan en la lógica difusa y otras técnicas de incertidumbre. Algunos ejemplos son:

- **Controladores de Tráfico:** En el control del tráfico y semáforos, se pueden utilizar sistemas de razonamiento impreciso para optimizar los tiempos de espera de los vehículos y mejorar el flujo del tráfico. La lógica difusa permite ajustar los tiempos de semáforos en función del flujo vehicular en tiempo real.
- **Diagnóstico Médico:** En medicina, se pueden aplicar sistemas de razonamiento impreciso para evaluar síntomas y proporcionar diagnósticos preliminares o sugerencias de tratamiento. Por ejemplo, en el diagnóstico de enfermedades como el cáncer, donde los resultados de las pruebas pueden no ser definitivos, los sistemas de razonamiento impreciso pueden ayudar a proporcionar una evaluación más completa y considerar múltiples factores para el diagnóstico.

El razonamiento impreciso es especialmente valioso cuando se enfrentan problemas en los que la información es vaga o incierta, lo que permite tomar decisiones más robustas y flexibles.

A continuación, un diagrama para ilustrar cómo un sistema de razonamiento impreciso maneja la incertidumbre en la toma de decisiones:



SISTEMAS BASADOS EN REGLAS

Los sistemas basados en reglas son sistemas de Inteligencia Artificial que utilizan reglas lógicas para representar el conocimiento y tomar decisiones. Cada regla consiste en una condición y una acción, y cuando se cumple la condición, se aplica la acción correspondiente. Algunos ejemplos son:

- **Sistemas de Recomendación:** Los sistemas de recomendación utilizan reglas lógicas para sugerir productos, películas, música u otros elementos en función del comportamiento del usuario y otros datos relevantes. Por ejemplo, plataformas de comercio electrónico como Amazon utilizan sistemas de recomendación basados en reglas para ofrecer productos relacionados basados en el historial de compras del usuario.
- **Diagnóstico en Sistemas de Soporte Médico:** En sistemas de soporte médico, las reglas se utilizan para evaluar síntomas y datos médicos y proporcionar diagnósticos preliminares o sugerencias de tratamiento. Por ejemplo, en sistemas de asistencia médica remota, las reglas basadas en síntomas pueden proporcionar recomendaciones iniciales antes de que el paciente sea atendido por un profesional de la salud.

Los sistemas basados en reglas son ampliamente utilizados en aplicaciones donde se requiere un razonamiento transparente y fácil de entender, ya que las reglas lógicas son explícitas y pueden ser interpretadas por humanos.

A continuación, un diagrama para ilustrar cómo un sistema basado en reglas aplica las reglas lógicas para tomar decisiones:



Estos modelos de sistemas de Inteligencia Artificial representan diferentes enfoques para resolver problemas y tomar decisiones en diversas aplicaciones. Cada uno de estos modelos tiene sus propias ventajas y desafíos, y la elección del enfoque adecuado depende del contexto y los requisitos específicos del problema a resolver.

Técnicas de la Inteligencia Artificial

A continuación veremos una taxonomía de técnicas que se usan en IA, algunas se estudiarán a fondo en este módulo (Modelos de Inteligencia Artificial - MIA) y otros en otro módulo del curso (Sistemas de Aprendizaje Automático - SAA).

Modelo Clásico. Sistemas expertos (MIA)

La Inteligencia Artificial, inicialmente, tuvo un desarrollo más teórico que práctico. Los planteamientos originarios de esta Inteligencia Artificial clásica se definieron para un tipo de trabajo informático que ignoraba en buena medida cómo se ha desarrollado en los últimos decenios y que actualmente está establecido como convencional.

Recuerda que estamos hablando de los años 60 del Siglo XX, y que en esa época apenas existían ordenadores experimentales, con una memoria y capacidad de cómputo que ahora consideraríamos ridículos. Cualquier Smartwatch o controlador de aspiradora inteligente tiene más memoria y velocidad de cálculo que los ordenadores que se utilizaron o se previó que se podrían utilizar para desarrollar Inteligencia Artificial entonces.

Otro aspecto importante a tener en cuenta sobre lo que se entendía por Inteligencia Artificial en esos primeros años es que se preveía que en un plazo de tiempo razonable iba a ser posible que las máquinas "pensaran" como los humanos. Es decir, que los mecanismos de la Inteligencia Artificial imitarían la manera de aprender y reaccionar (actuar) del cerebro humano.

Se dedicó tiempo y esfuerzo, por tanto, a intentar definir de manera matemática y computacional cómo funcionaba el cerebro humano. En última instancia se intentó definir un proceso informático (basado en algoritmos matemáticos) equivalente a lo que haría una neurona humana.

Por tanto, para entender bien qué es la Inteligencia Artificial Clásica, debemos tener en cuenta que:

- Lo que ahora denominamos **Inteligencia Artificial Clásica** fue más bien un **ejercicio de creación de principios generales**, que posteriormente se emplearon para desarrollar los primeros programas informáticos prácticos de Inteligencia Artificial aplicada. Pero esta IA está bastante alejada de lo que hoy por hoy entendemos a nivel práctico como Inteligencia Artificial.

- La Inteligencia Artificial Clásica **quería desarrollar programas informáticos que replicaran el conocimiento humano**, inicialmente en casos particulares y "sencillos", con la intención de ir poco a poco abarcando procesos y casos más complejos. De tal manera que la máquina pudiera "pensar" y actuar como un humano experto en dicho caso particular.

La Inteligencia Artificial Clásica necesitaba que en el proceso de aprendizaje de dicha IA participaran "expertos" en la tarea que se pretendía que la máquina realizará por sí misma.

Si se quería que una máquina aprendiera a jugar al ajedrez, en el proceso de aprendizaje era necesario contar con expertos jugadores de ajedrez. De esa necesidad de contar con "expertos" se acabó extendiendo el término "**Sistema Experto**" para designar a los primeros programas de IA que se desarrollaron.

Siendo más concretos, la definición de Sistema Experto es:

Un sistema experto es un programa informático que se ha desarrollado a partir de nuestro conocimiento sobre una cuestión, y que consigue que el ordenador muestre un comportamiento equivalente al que tendría un experto humano sobre el mismo tema

En esencia se seguía un proceso con cuatro fases:

1. **Localizar al humano experto con conocimiento:** Según el caso particular para el que se quisiera crear esa IA, era necesario incorporar al equipo de desarrollo a una o varias personas expertas en la materia, para que aportaran todo el conocimiento en profundidad.
2. **Definir reglas:** Ese conocimiento humano había que convertirlo en reglas lo más sencillas posible, que relacionaran los diferentes casos y aspectos del conocimiento que se pretendía replicar con la IA.
3. **Informatizar:** Esas reglas había que traducirlas a lenguaje informático.
4. **Iterar:** Probar a ver si realmente la máquina se comportaba de forma "inteligente", buscar fallos, redefinir reglas, o mejorar la programación, y volver a probar. Así tantas veces como fuera necesario hasta que se pudiera considerar que la máquina actuaba igual de bien que el experto humano.

La Inteligencia Artificial Clásica quería "informatizar" modelos de conocimiento. Es decir, lograr convertir en programas informáticos capacidades humanas como "jugar al ajedrez", "detectar faltas de ortografía", "aprender un idioma"...

Pero esta manera de programar Inteligencia Artificial tiene bastantes limitaciones. **Sólo es asequible cuando el conocimiento o "inteligencia" que se quiere informatizar se basa en una relación de causalidad: Causa-Efecto.**

En el caso del **juego del ajedrez**:

1. Se busca a una o varias personas expertas jugadoras de ajedrez, que conozcan en profundidad el juego, sepan cuáles son las jugadas más características, etc.
2. Con la ayuda de estos expertos jugadores, los científicos definen todos los aspectos del juego de ajedrez, desde cómo es el tablero, las fichas, los movimientos, la jerarquía o relación de importancia entre las fichas, las posibles reacciones a los movimientos del contrario.
3. Los informáticos toman todas esas reglas y las traducen a lenguaje de programación.
4. Se comprueba que el ordenador sea capaz de jugar al ajedrez, sin equivocarse, y priorizando los movimientos que antes le permitan obtener la victoria. Si algo sale mal o se detectan fallos, hay que volver a revisar todo el proceso y mejorarlo (redefinir la forma de algunas normas, o la programación, etc).

Cada posible movimiento del contrario permite que la Inteligencia Artificial reaccione de diferentes maneras (moviendo tal o cual ficha). A su vez, este movimiento de la Inteligencia Artificial permite otras diferentes maneras de reaccionar por parte del contrario... Son lo que hemos mencionado más arriba: relaciones de causalidad. Cada acción tiene una serie de posibles reacciones (y la IA debe elegir una de ellas), que a su vez tienen otra serie de reacciones posibles (el contrario debe elegir una) y así sucesivamente. Gracias a la memoria de la computadora y la capacidad de cómputo es capaz de ver todas las posibles situaciones a 10, 20, 30... movimientos; e ir escogiendo los movimientos que con mayor probabilidad le puedan llevar a la victoria.

Cuando el conocimiento o "inteligencia" que se quiere informatizar se basa en una correlación (relaciones proporcionales entre todas las variables que intervienen) es prácticamente imposible definir y traducir a lenguaje informático todas esas reglas y relaciones. Para estos casos necesitamos otra manera de abordar la Inteligencia Artificial... que es la que se ha desarrollado posteriormente y veremos en los siguientes apartados.

Aprendizaje Automático (Machine Learning) (SAA)

El **Aprendizaje Automático (Machine Learning)** es una rama clave de la Inteligencia Artificial que permite a las máquinas aprender y mejorar su rendimiento en tareas específicas a través de la experiencia. En lugar de ser programadas explícitamente para realizar una tarea, las máquinas utilizan datos para aprender patrones y tomar decisiones informadas. El Aprendizaje

Automático se ha convertido en una herramienta poderosa en una variedad de aplicaciones, desde reconocimiento de voz y visión por computadora hasta recomendaciones de productos y diagnósticos médicos.

Es importante entender que el Aprendizaje Automático es una rama de la IA, aunque en la actualidad ha adquirido mucha importancia y se utiliza en prácticamente todos los proyectos de IA. De manera que hoy cuando hablamos de Inteligencia Artificial en realidad estamos hablando de esta rama concreta (**el todo por la parte**).

El Machine Learning o Aprendizaje Autónomo (Automático) a su vez ha evolucionado en estos pocos años que lleva desarrollándose: Inicialmente se focalizaba en lograr que la máquina aprendiera basándose en datos, a través de estudiar el reconocimiento de patrones (casos similares entre el total de elementos del data set o base de datos). Actualmente se centra más bien en "resolver" problemas prácticos que en "aprender", aunque evidentemente "aprende" (pero el aprendizaje como tal ya no es el foco, sino el resultado obtenido). Al reconocimiento de patrones que ya se usaba desde el principio añade ahora lo que conocemos como el razonamiento probabilístico, la estadística y la recuperación de datos.

DEFINICIONES DE APRENDIZAJE AUTOMÁTICO

Arthur Samuel (que trabajó para IBM) en 1959 describía el Aprendizaje Automático como:

El campo del estudio que da a las computadoras la capacidad de aprender sin ser programadas explícitamente

Esta es una definición antigua e informal respecto a lo que hoy en día entendemos por Machine Learning.

Tom Mitchell (profesor en la Universidad de Carnegie Mellon) ha ofrecido una definición más moderna:

Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y medida de rendimiento P, si su desempeño en las tareas en T medido por P mejora con la experiencia E

Jugar a las damas.

- E es la experiencia de jugar muchas partidas de damas.
- T es la tarea de jugar a las damas.
- P es la probabilidad de que el programa gane la partida actual.

A medida que la máquina "observa" el desarrollo de cada partida gana experiencia. Gracias a esta experiencia acaba siendo capaz de realizar la tarea (jugar a las damas) por sí misma. Y además va comprobando el rendimiento obtenido en cada partida (si gana o no gana, en cuántos movimientos, etc), por lo que va perfeccionando su capacidad de jugar de manera eficaz.

El Aprendizaje Automático consiste en un programa informático que analiza y aprende de los datos que le proporcionamos para decidir qué hacer con ellos y proporcionar respuestas. Genera reglas para, con eso que ha "aprendido", acelerar procesos, reconocer patrones, segmentar grupos (personas, hábitos, etc). Lo fundamental es que el "cómo aprende" es automático; nosotros sólo le tenemos que dar datos o ejemplos de partida.

La definición de Aprendizaje Automático más aproximada a lo que entendemos actualmente sería:

El Aprendizaje Automático (Machine Learning) es un proceso de adquisición de conocimiento de manera automática mediante la utilización de ejemplos (experiencia) de entrenamiento.

TIPOS DE APRENDIZAJE AUTOMÁTICO

Aprendizaje Supervisado

La característica fundamental del Aprendizaje Automático Supervisado es que dicho aprendizaje se realiza **a partir de datos que ya han sido etiquetados previamente**.

¿Qué queremos decir con datos etiquetados? Pues que al programa que va a "aprender" le proporcionamos los datos indicando sus características (bien las de entrada, bien las de salida). Por ejemplo, si queremos que un programa de IA sea capaz de distinguir en qué fotos aparece un perro, al proporcionarle fotos para el aprendizaje (datos de entrada) ya le decimos en cuáles aparecen gatos, en cuáles perros y en cuáles patos... Podemos decir que "supervisamos" el aprendizaje dándole pistas al programa de Inteligencia Artificial.

En realidad el término correcto que debemos emplear es el de instancias: que son cada uno de los elementos que forman el conjunto de datos (en el ejemplo, cada foto), se componen de una serie de campos de características o atributos (en el ejemplo, aparecer gato, aparecer perro, aparecer pato...) y un campo objetivo (en el ejemplo, aparecer perro), que es el que se encuentra etiquetado en los datos de entrenamiento. El objetivo de este tipo de aprendizaje es extraer un conjunto de reglas que permitan predecir el campo objetivo para nuevos casos de estudio.

Los problemas de Aprendizaje Supervisado **se dividen en dos categorías: Regresión y Clasificación**. La diferencia entre estas dos categorías radica en el tipo de campo objetivo (lo que queremos que la Inteligencia Artificial nos dé como respuesta), que es numérico en el caso de la Regresión y categórico en el caso de la Clasificación.

Aprendizaje no Supervisado

En este tipo de aprendizaje **no se requiere un etiquetado previo** de las instancias, pues **el objetivo es encontrar relaciones de similitud, diferencia o asociación** en el conjunto de datos.

Es decir, que no "le decimos" a la Inteligencia Artificial qué estamos buscando, ni cuál es el dato concreto sobre el que queremos que haga una predicción. Asumimos que hay ciertos tipos de relación y dependencias entre los diversos datos, pero queremos que sea la Inteligencia Artificial la que encuentre esas relaciones. En muchas ocasiones nos llevamos sorpresas, cuando la IA nos muestra semejanzas entre datos que nos han pasado desapercibidas a los humanos.

Como hemos dicho, el objetivo es que la IA encuentre relaciones de tres tipos:

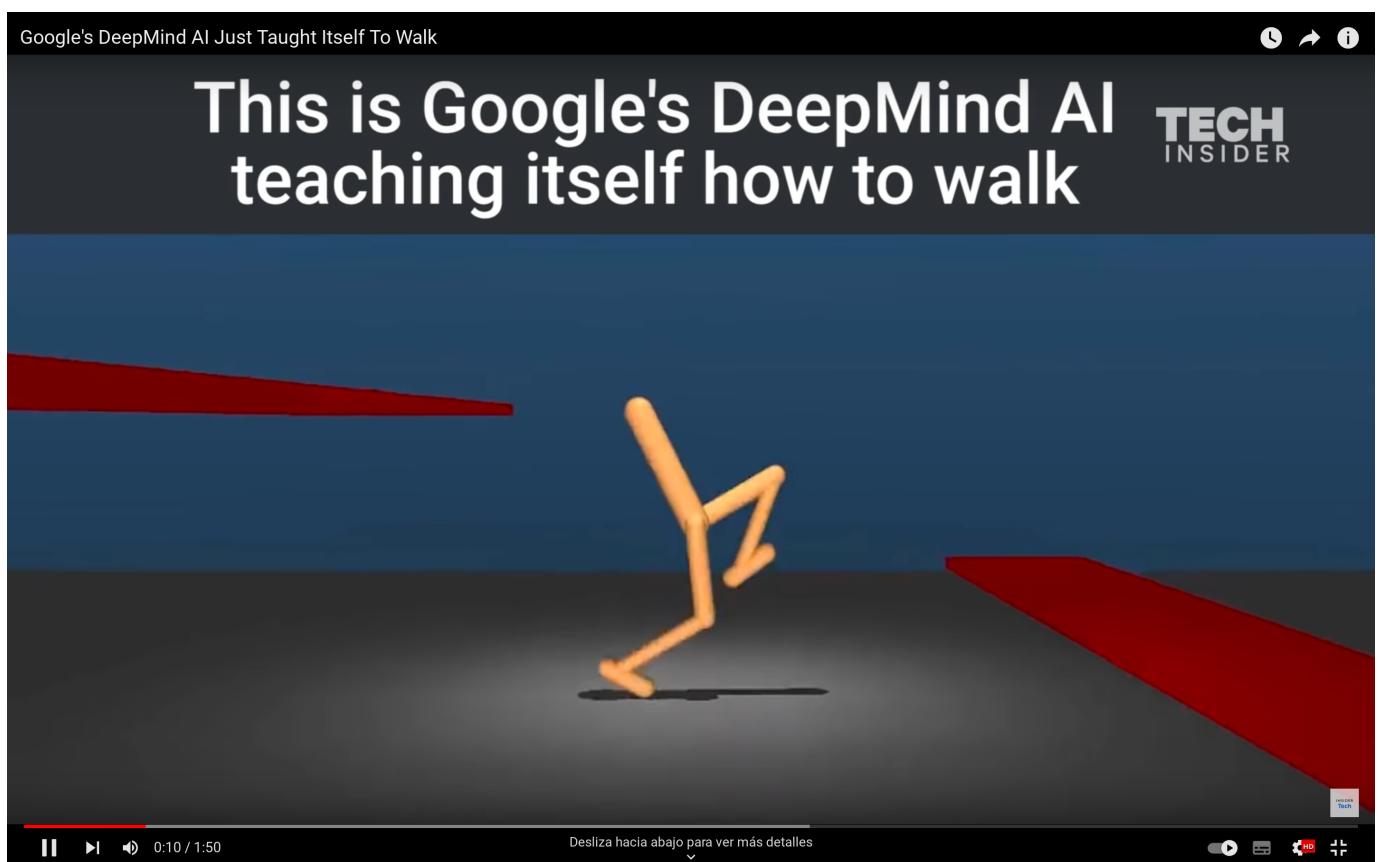
- Similitudes.
- Diferencias.
- Asociaciones.

Aprendizaje por Refuerzo

Existe también el Aprendizaje por Refuerzo, en el que **el objetivo es aprender cómo mapear situaciones o acciones para maximizar una cierta recompensa**. Se trata de programar agentes mediante premio y castigo sin necesidad de especificar cómo realizar la tarea.

Uno de los casos más conocidos de refuerzo automático es el cuando en la empresa Deep Mind lograron "enseñar" a jugar al Arkanoid (¿Te acuerdas? lo vimos en el punto [Historia de la IA](#)). Lo hicieron con este modelo de Aprendizaje. La IA solo conocía los parámetros básicos de movimiento, y los "premios" (puntos por romper bloques, puntos por tardar lo menos posible en terminar la partida) y "castigos" (finalizar la partida sin puntos si se perdía la pelota por el extremo inferior de la pantalla).

En este tipo de problemas lo más importante es definir y programar las condiciones que deben cumplirse (las reglas del juego, qué se puede hacer y cómo interactúan unos elementos con otros). Por ejemplo, en el siguiente vídeo, podemos ver cómo una serie de personajes digitales han "aprendido" a caminar, correr y sortear obstáculos. Se ve claramente que los programadores han sido precisos para que "los brazos" se mantengan articulados al cuerpo, igual que las "patas". Pero no parece que hayan especificado mucho sobre la gravedad o sobre "el cansancio" que supone correr con los brazos hacia arriba.



<https://www.youtube.com/watch?v=gn4nRCC9TwQ>

Redes Neuronales Artificiales (SAA y PIA)

CONCEPTOS BÁSICOS Y FUNCIONAMIENTO

Las redes neuronales artificiales están inspiradas en la estructura y funcionamiento del cerebro humano.

Consisten en una colección de nodos interconectados (neuronas artificiales) organizados en capas que transmiten señales entre ellas. Cada neurona recibe entradas ponderadas, las procesa mediante una función de activación y produce una salida que se envía a otras neuronas.

El proceso de aprendizaje en las redes neuronales se basa en ajustar los pesos de las conexiones entre las neuronas para que el modelo pueda realizar predicciones precisas y generalizadas en nuevos datos.

APLICACIONES Y ARQUITECTURAS COMUNES

Las redes neuronales artificiales tienen una amplia gama de aplicaciones, algunas de las cuales incluyen:

- **Reconocimiento de Patrones:** Clasificación de datos en diferentes categorías, como en reconocimiento de imágenes y diagnóstico médico.
- **Procesamiento de Lenguaje Natural:** Análisis de texto y generación de texto coherente, como en traducción automática y generación de subtítulos.
- **Juegos y Control de Robots:** Entrenamiento de agentes para jugar juegos y controlar robots mediante técnicas de aprendizaje por refuerzo.

Las arquitecturas comunes de redes neuronales incluyen:

- **Redes Neuronales Feedforward:** Son las más básicas, donde las señales solo se transmiten en una dirección, desde la entrada hasta la salida, sin ciclos.
- **Redes Neuronales Recurrentes:** Tienen conexiones cíclicas que permiten el procesamiento de secuencias de datos, lo que las hace adecuadas para tareas de procesamiento de lenguaje natural y series de tiempo.

Algoritmos Genéticos

PRINCIPIOS BÁSICOS Y FUNCIONAMIENTO

Los algoritmos genéticos son una clase de algoritmos de computación evolutiva inspirados en el proceso de selección natural. Utilizan principios biológicos como la reproducción, mutación y selección para buscar soluciones óptimas en problemas de optimización y búsqueda heurística.

En un algoritmo genético, se crea una población inicial de soluciones candidatas, y luego se evalúa su aptitud en función de una función objetivo. Las soluciones con mayor aptitud tienen una mayor probabilidad de ser seleccionadas para reproducirse y producir descendencia mediante operaciones de cruce y mutación. Este proceso se repite a lo largo de generaciones, buscando converger hacia una solución óptima.

OPTIMIZACIÓN Y BÚSQUEDA HEURÍSTICA

Los algoritmos genéticos son ampliamente utilizados en problemas de optimización y búsqueda heurística, como:

- **Problemas de Optimización:** Encontrar la mejor solución posible en un espacio de búsqueda grande, como en el diseño de redes de transporte, rutas de vehículos y programación de horarios.
- **Diseño y Aprendizaje de Parámetros:** Optimización de parámetros en modelos de aprendizaje automático y redes neuronales.

Lógica Difusa

FUNDAMENTOS Y USO EN SISTEMAS DE TOMA DE DECISIONES

La lógica difusa es una extensión de la lógica clásica que permite manejar incertidumbre y vaguedad en los datos. A diferencia de la lógica binaria (verdadero/falso), la lógica difusa utiliza grados de verdad entre 0 y 1, lo que permite representar y razonar con conceptos imprecisos.

La lógica difusa es especialmente útil en sistemas de toma de decisiones, donde las condiciones y resultados pueden ser vagos o subjetivos. Los conjuntos difusos y las reglas de inferencia difusa se utilizan para modelar y resolver problemas con datos inciertos.

VENTAJAS Y DESVENTAJAS FRENTE A LA LÓGICA CLÁSICA

Las ventajas de la lógica difusa incluyen:

- **Tratamiento de Incertidumbre:** Permite manejar datos imprecisos o ambiguos en un contexto más cercano a la forma en que los humanos toman decisiones.
- **Adaptabilidad:** Es útil para modelar sistemas complejos y no lineales donde las relaciones son difíciles de expresar con precisión.

Sin embargo, también tiene algunas desventajas:

- **Complejidad Computacional:** El procesamiento de la lógica difusa puede ser más complejo y costoso en términos computacionales en comparación con la lógica clásica.
- **Interpretación de Resultados:** La interpretación de los resultados difusos puede ser subjetiva y depender del contexto, lo que dificulta la comparación entre diferentes sistemas.

Campos de Aplicaciones de la Inteligencia Artificial

Visión por Computadora

La visión por computadora es un campo de la inteligencia artificial que se enfoca en **enseñar a las máquinas a interpretar y comprender el mundo visual**, permitiéndoles analizar y procesar imágenes y videos. Esta área ha experimentado un rápido avance en los últimos años gracias a los avances en técnicas de Aprendizaje Profundo y el aumento de la capacidad computacional. La visión por computadora tiene una amplia gama de aplicaciones en diversos campos, desde la medicina y la industria hasta la seguridad y el entretenimiento.

Los nuevos desarrollos de reconocimiento de imagen y visión artificial no han tenido un origen único y concreto, sino que se han ido conformando por la aportación de investigadores e ingenieros que han compartido sus ideas, como es el caso de **Yann Lecun**, que ideó **LeNet** usando, ya a **finales 90**, redes convolucionales para el reconocimiento de dígitos manuscritos.

El lanzamiento de **ImageNet**, abierto y gratuito, por parte de **Fei Fei Li**, que ya ha alcanzado más de 14 millones de imágenes, supuso un gran impulso al desarrollo de nuevas aplicaciones de visión artificial.

Tú mismo puedes descargarte un dataset reducido (¡aunque es de 166 GB!) para probar en [Kaggle](#).

Cada vez más y mejores datasets de imágenes etiquetadas, y un mayor conocimiento y dominio de redes convolucionales, han revolucionado el campo de la visión computacional, que ha pasado de ser una cuestión de más resolución o renderizados 3D, a una cuestión más cognitiva. Se ha conseguido crear modelos que realmente entienden qué están viendo.

La visión artificial automatiza la extracción, el análisis, la clasificación y la comprensión de la información útil a partir de los datos de las imágenes. Los datos de la imagen adoptan muchas formas, como las siguientes:

- Imágenes individuales
- Secuencias de video
- Visualizaciones de varias cámaras
- Datos tridimensionales

APLICACIONES DE VISIÓN POR COMPUTADORA

Vigilancia

La utilización de cámaras para sistemas de vigilancia y seguridad, se ha extendido de forma generalizada en nuestra sociedad actual. Pero, en algunos casos, estas cámaras tienen el plus de formar parte de un sistema de inteligencia artificial.

Esta aplicación de los sistemas de reconocimiento de imagen son bastante controvertidos, pues plantean muchas dudas éticas respecto a la libertad fundamental de las personas. Es una herramienta muy útil y potente, y puede servir para beneficiar al ser humano tanto como para perjudicarlo. Lo bueno es que la comunidad en torno a la inteligencia artificial va descubriendo e ideando formas sencillas de evitar o combatir usos deshonestos de este tipo de herramientas.

En la mayoría de los casos, el software hace cálculos agregados de lo que "ve" y devuelve valores de ciertos indicadores, en vez de la imagen o fragmento de grabación con los datos personales. Solo en países donde hay regímenes autoritarios se mantienen prácticas que atentan contra los derechos de los ciudadanos.

Entre las aplicaciones concretas de este tipo de sistemas, a parte de las obvias por parte de la policía o sistemas de seguridad de organizaciones, están:

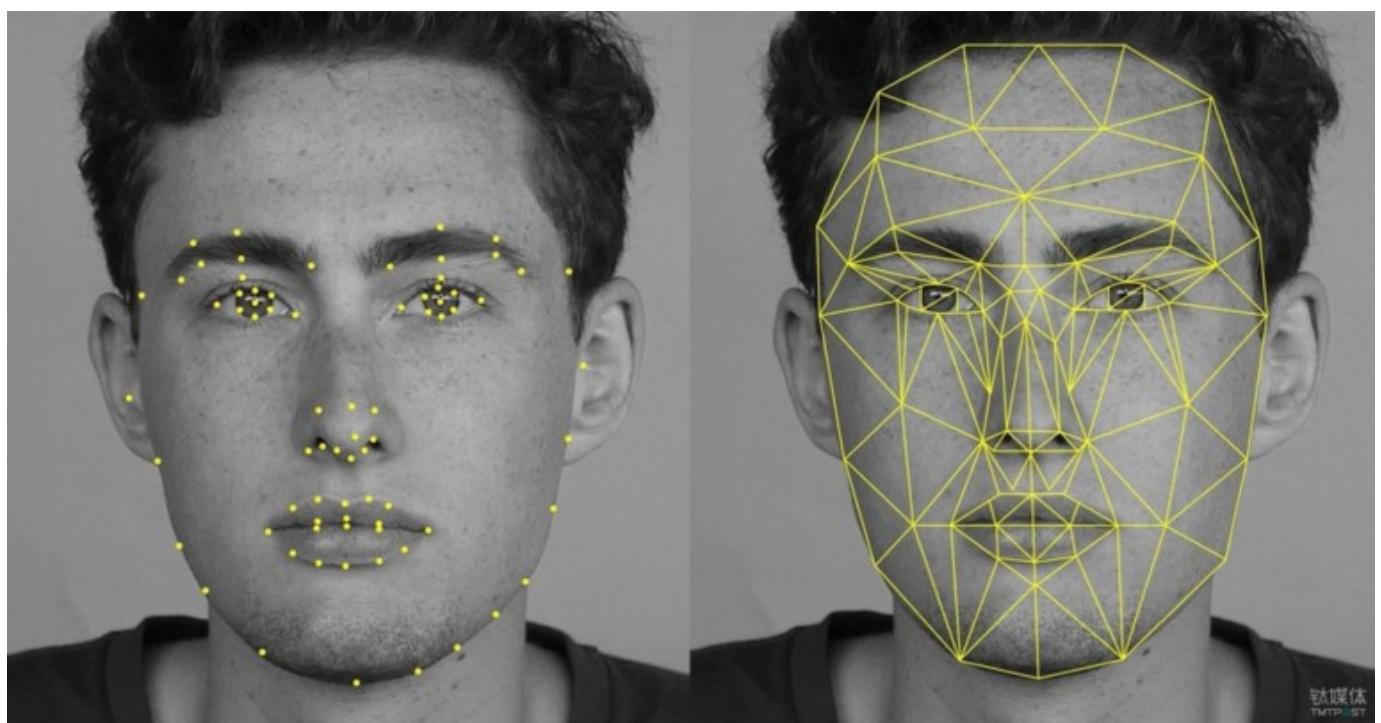
- Vigilancia y control del tráfico en las ciudades.
- Cuidado de personas mayores.
- Detección de infracciones de reglas sanitarias en la industria (especialmente en la industria alimentaria).
- Monitorización del uso de infraestructuras críticas o adscritas a normas de utilización.

- Monitorización de funcionamiento y estados en líneas de producción.
- Esto son solo algunos ejemplos, pero cualquier proceso o sistema en el cual se puedan detectar anomalías a través de la imagen, sería un buen candidato para aplicar este tipo de solución.

Incluso la inteligencia artificial puede ayudar a hacer más respetuosas con la privacidad ciertas aplicaciones y herramientas que ya se estaban utilizando, como el caso del software [Cherry Home de la empresa AvantGuard](#).

Reconocimiento Facial

Un analizador facial es un software que identifica o confirma la identidad de una persona a partir del rostro. Funciona mediante la identificación y medición de los rasgos faciales en una imagen. El reconocimiento facial puede identificar rostros humanos en imágenes o videos, determinar si el rostro que aparece en dos imágenes pertenece a la misma persona o buscar un rostro entre una gran colección de imágenes existentes. Los sistemas de seguridad biométricos utilizan el reconocimiento facial para identificar de forma exclusiva a las personas durante la incorporación o el inicio de sesión de los usuarios, así como para reforzar la actividad de autenticación de estos. Los dispositivos móviles y personales también utilizan con frecuencia la tecnología de los analizadores faciales para proteger los dispositivos.



Se pueden detectar los datos faciales tanto en los perfiles frontales como en los laterales del rostro. El sistema de reconocimiento facial analiza la imagen del rostro. Asigna y lee la geometría del rostro y las expresiones faciales. Identifica los puntos de referencia faciales que son clave para distinguir un rostro de otros objetos. La tecnología de reconocimiento facial por lo general busca lo siguiente:

- Distancia entre los ojos
- Distancia de la frente a la barbilla
- Distancia entre la nariz y la boca
- Profundidad de las cuencas oculares
- Forma de los pómulos
- Contorno de los labios, las orejas y la barbilla

El sistema convierte los datos de reconocimiento facial en una cadena de números o puntos denominada huella facial. Cada persona tiene una huella facial única, de forma similar a una huella dactilar. La información utilizada por el reconocimiento facial también se puede utilizar a la inversa para reconstruir digitalmente el rostro de una persona.

El reconocimiento facial puede identificar a una persona al comparar los rostros de dos o más imágenes y evaluar la probabilidad de que coincidan. Por ejemplo, puede verificar que el rostro mostrado en una autofoto tomada con la cámara de un móvil coincide con el rostro de una imagen de un documento de identidad emitido por el gobierno, como un permiso de conducir o un pasaporte, así como verificar que el rostro que aparece en la autofoto no coincide con un rostro de un conjunto de rostros capturados previamente.

Aplicación de Visión por Computadora:

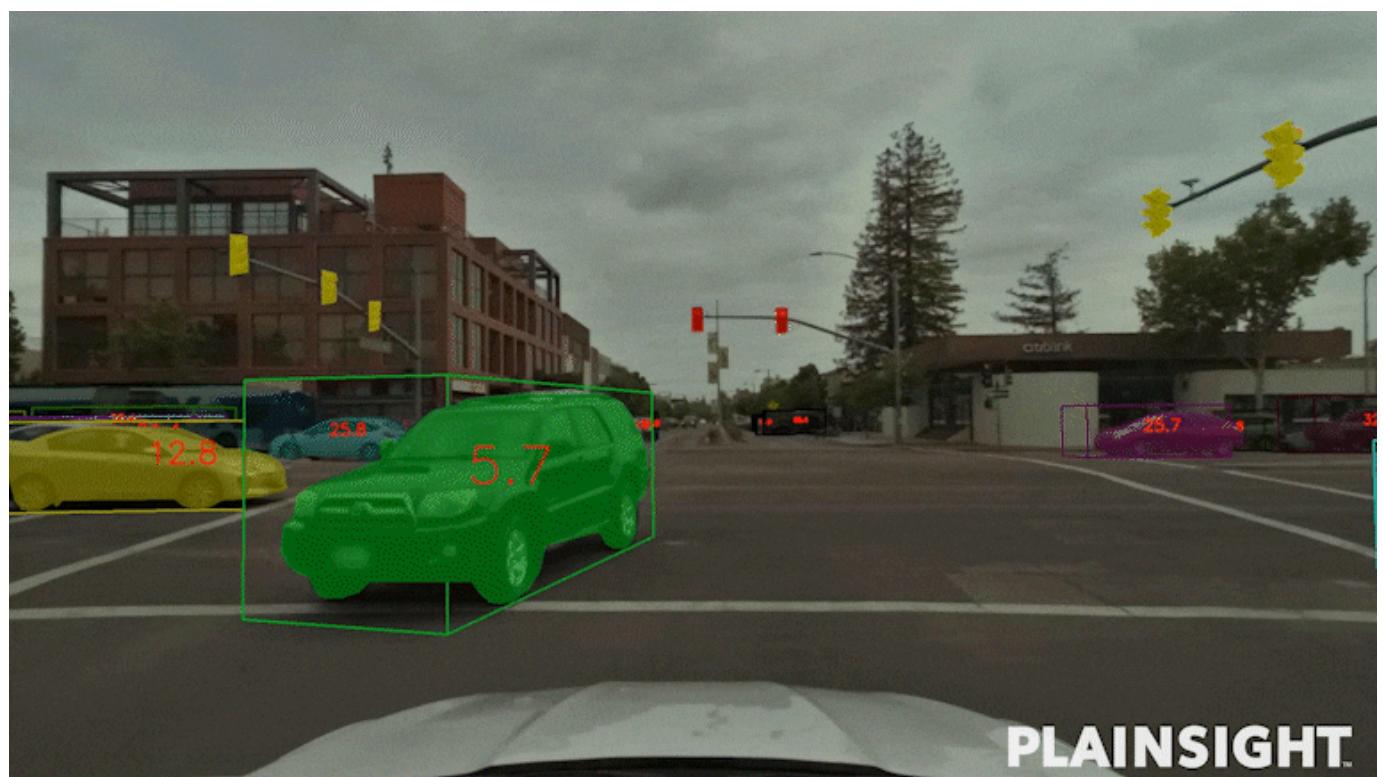
Un ejemplo práctico de aplicación de visión por computadora es el reconocimiento facial utilizado en aplicaciones de seguridad y desbloqueo de dispositivos. En este caso:

- **Entrada:** La entrada es una imagen o un video que contiene rostros humanos.
- **Procesamiento de Imagen:** El sistema de visión por computadora procesa la imagen para detectar y extraer características clave del rostro, como ojos, nariz, boca, etc.
- **Aprendizaje Automático:** Las características del rostro se utilizan como entrada para un modelo de aprendizaje automático previamente entrenado. El modelo clasifica las características y compara con una base de datos de rostros previamente almacenados.
- **Salida:** Como resultado, el sistema identifica o verifica la identidad del individuo y permite el acceso o desbloqueo según los resultados.

Conducción autónoma.

El sistema de conducción autónoma de vehículos implica varias tareas y subsistemas, pero uno de los más importantes, es el de visión artificial, pues la mayoría de las decisiones de seguridad del coche se basan en lo que captan las cámaras.

La cuestión crítica en estos sistemas, es el reconocimiento de señales de tráfico u objetos/obstáculos alrededor del vehículo a una velocidad relativamente alta. Por ejemplo, si el coche debe parar porque hay una persona cruzando la carretera, el sistema de visión debe captar la imagen con antelación suficiente como para frenar a una distancia también suficiente.



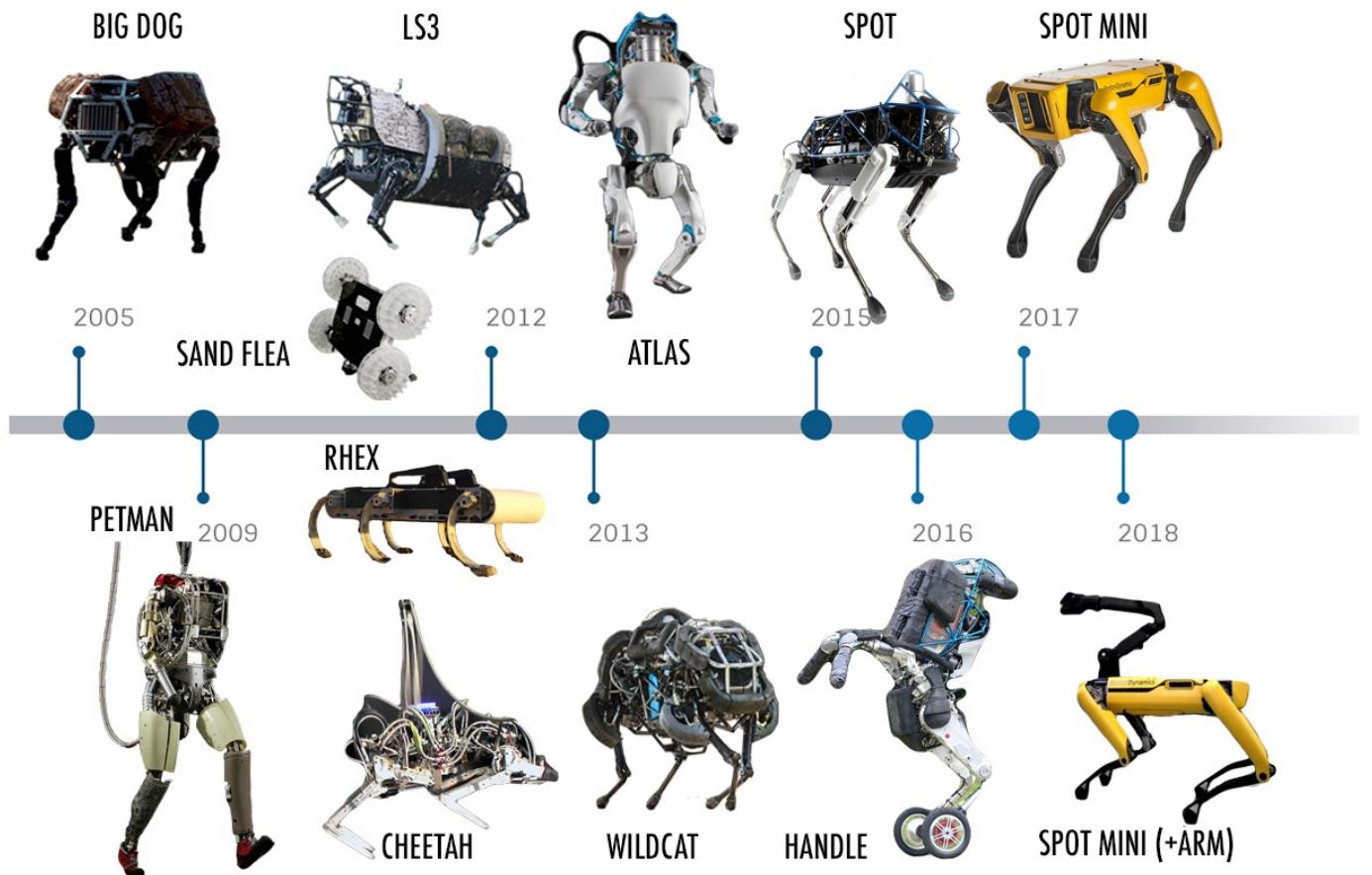
Captar cómo son las líneas de la carretera para ir girando lo que corresponda, también requiere ir captando esas variaciones de trayectoria suficientemente rápido, pues en carretera es muy común ir a velocidades altas. De hecho, hay sistemas que, a partir de cierta velocidad, no permiten usar la función de conducción autónoma.

Si quieras conocer mejor la clasificación de niveles de conducción autónoma y cómo se relaciona el ámbito de la visión artificial con ellos, te recomendamos leer el artículo "[Autonomous Vehicles Are Driving Computer Vision Into the Future](#)".

Sistema auxiliar en robots.

Los robots son sistemas complejos que suelen ejecutar una serie de tareas en el mundo físico en base a una secuencia programada. En la industria, se han estado utilizando sistemas robóticos desde hace muchos años. Pero este campo también ha ido evolucionando, y la inteligencia artificial está aportando grandes avances que causan un importante impacto en el alcance de estos sistemas.

BOSTON DYNAMICS



Un sistema robótico tiene tres partes fundamentales:

- Sensores o entradas.
- Sistema de control.
- Actuadores.

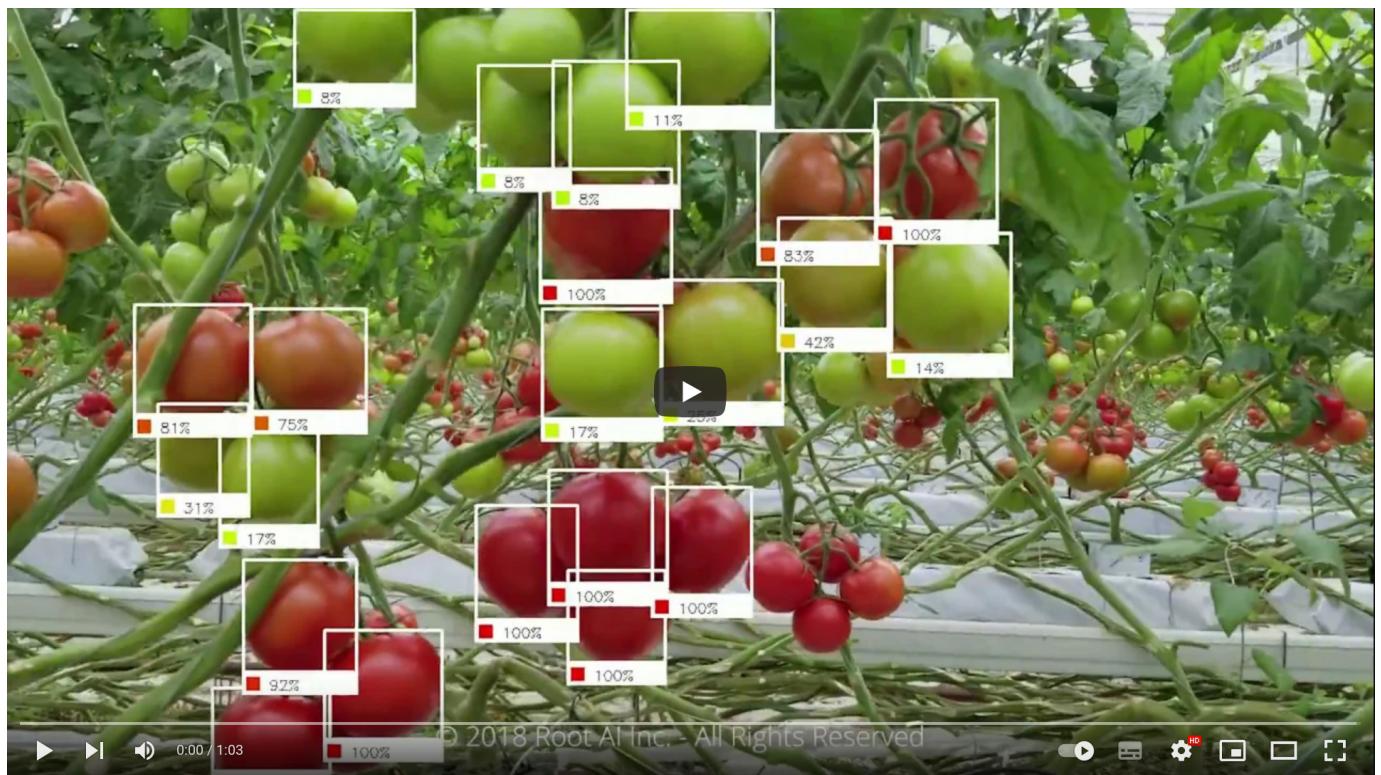
Mas adelante hablaremos de cómo los sistemas de control se han beneficiado de la inteligencia artificial, pero aquí nos detenemos en el módulo de visión artificial como parte del conjunto de sensores que aportan los estímulos o la información que el sistema robótico va a necesitar para la toma de decisiones.

El sistema de visión artificial de un robot, le permite detectar objetos y posicionarse a sí mismo o a objetos que transporta en función de lo que está viendo. Esto es un gran avance respecto a otros sistemas de posicionamiento, que exigían constantes tareas de calibración y ralentizaban las tareas del robot.

<https://bostondynamics.com/videos/>

Reconocimiento de Objetos

El reconocimiento de objetos implica identificar y localizar objetos específicos en imágenes o videos. Algunos ejemplos incluyen reconocimiento de vehículos en carreteras, detección de peatones en sistemas de asistencia al conductor y clasificación de objetos en aplicaciones de etiquetado automático.



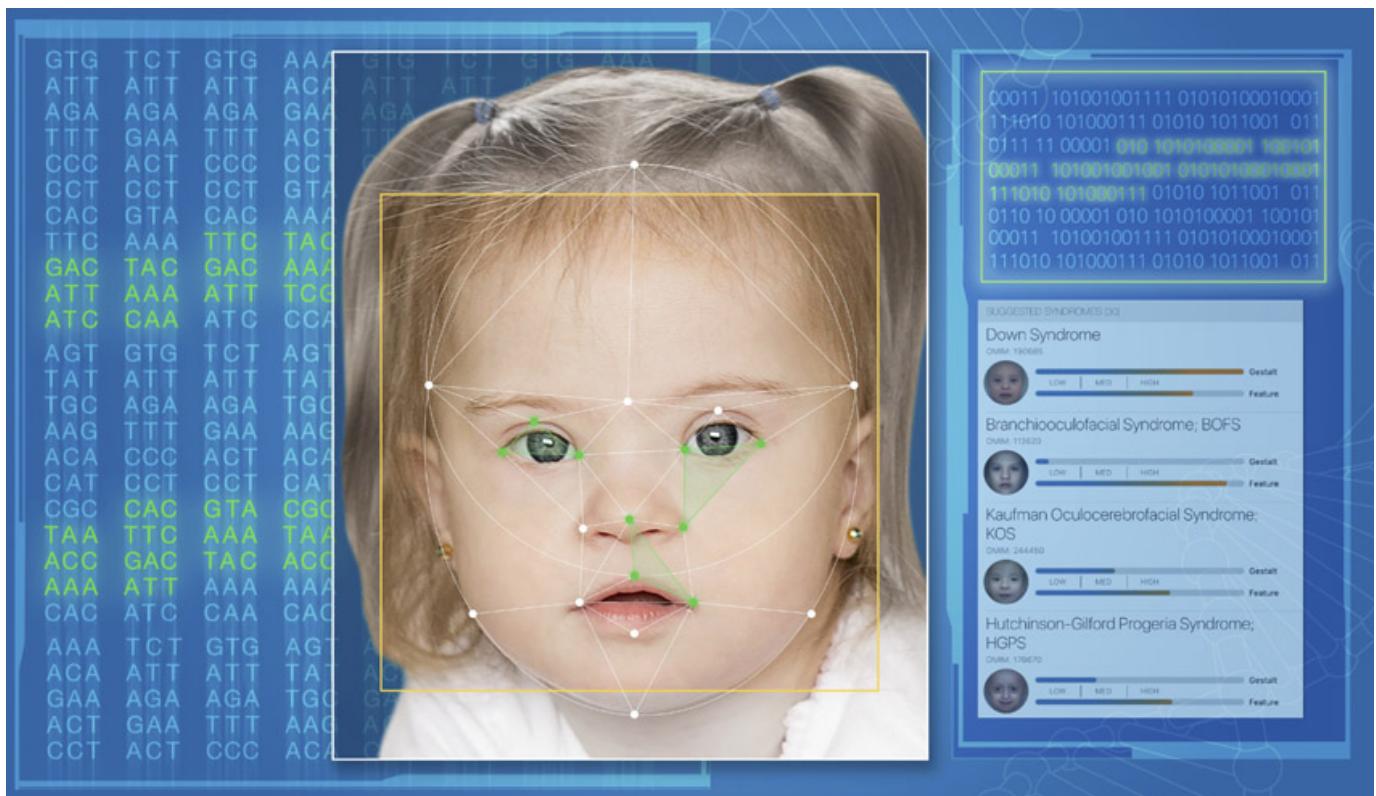
En la industria agroalimentaria, la capacidad de visión inteligente es de vital importancia, porque constituye una parte decisiva de cara a que se obtenga un buen producto o una buena cosecha. En algunos casos, el robot sabe distinguir, mejor que el humano, si una fruta está en su momento óptimo de cosecha.

<https://www.youtube.com/watch?v=c-JduOfLEpc>

Detección y diagnóstico

En el campo de la medicina, la inteligencia artificial está teniendo un impacto de muchísimo valor, porque no solo consigue mejorar la vida de las personas, es que, literalmente, en algunos casos consigue salvar vidas. Es el caso de herramientas de inteligencia artificial para la detección y diagnóstico de enfermedades a través de imágenes.

Existen muchos programas informáticos de apoyo y ayuda al diagnóstico que han ido mejorando su aprendizaje a través de su uso repetido y continuado. Actualmente existen diferentes tipos de software que se pueden aplicar a diferentes grupos de enfermedades como MYCIN/MYCIN II para enfermedades infecciosas, CASNET para oftalmología, PIP para enfermedades renales o Al/RHEUM para enfermedades reumatólogicas. La empresa FDNA a través de su software de reconocimiento facial Face2Gene® es capaz de apoyar o sospechar el diagnóstico de más de 8.000 enfermedades raras, con un reciente ensayo clínico desarrollado en Japón con buenos resultados.



En el campo del procesamiento y la interpretación de imágenes para el diagnóstico, la IA ofrece algoritmos que mejoran la calidad y la precisión del diagnóstico ya que los métodos de IA son excelentes para reconocer automáticamente patrones complejos en los datos de imágenes, elimina ruido en las imágenes ofreciendo una mayor calidad y permite establecer modelos tridimensionales a partir de imágenes de pacientes concretos.

Investigadores de IBM publicaron una investigación en torno a un nuevo modelo de IA que puede predecir el desarrollo del cáncer de mama maligno, con tasas comparables a las de los radiólogos humanos. Este algoritmo aprende y toma decisiones tanto de datos de imágenes como del historial de la paciente, pudo predecir correctamente el desarrollo del cáncer de mama en el 87% de los casos analizados, y también pudo interpretar el 77% de los casos no cancerosos. Este modelo podría algún día ayudar a los radiólogos a confirmar o negar casos positivos de cáncer de mama. Si bien los falsos positivos pueden causar una enorme cantidad de estrés y ansiedad indebidos, los falsos negativos a menudo pueden obstaculizar la detección temprana y el tratamiento posterior de un cáncer. Cuando se puso a prueba frente a 71 casos diferentes que los radiólogos habían determinado originalmente como «no malignos», pero que finalmente terminaron siendo diagnosticados con cáncer de mama dentro del año, el sistema de IA pudo identificar correctamente el cáncer de mama en el 48% de las personas (48% de los 71 casos), que de lo contrario no se habrían detectado (Fuente: "[La inteligencia artificial y sus aplicaciones en medicina](#)")

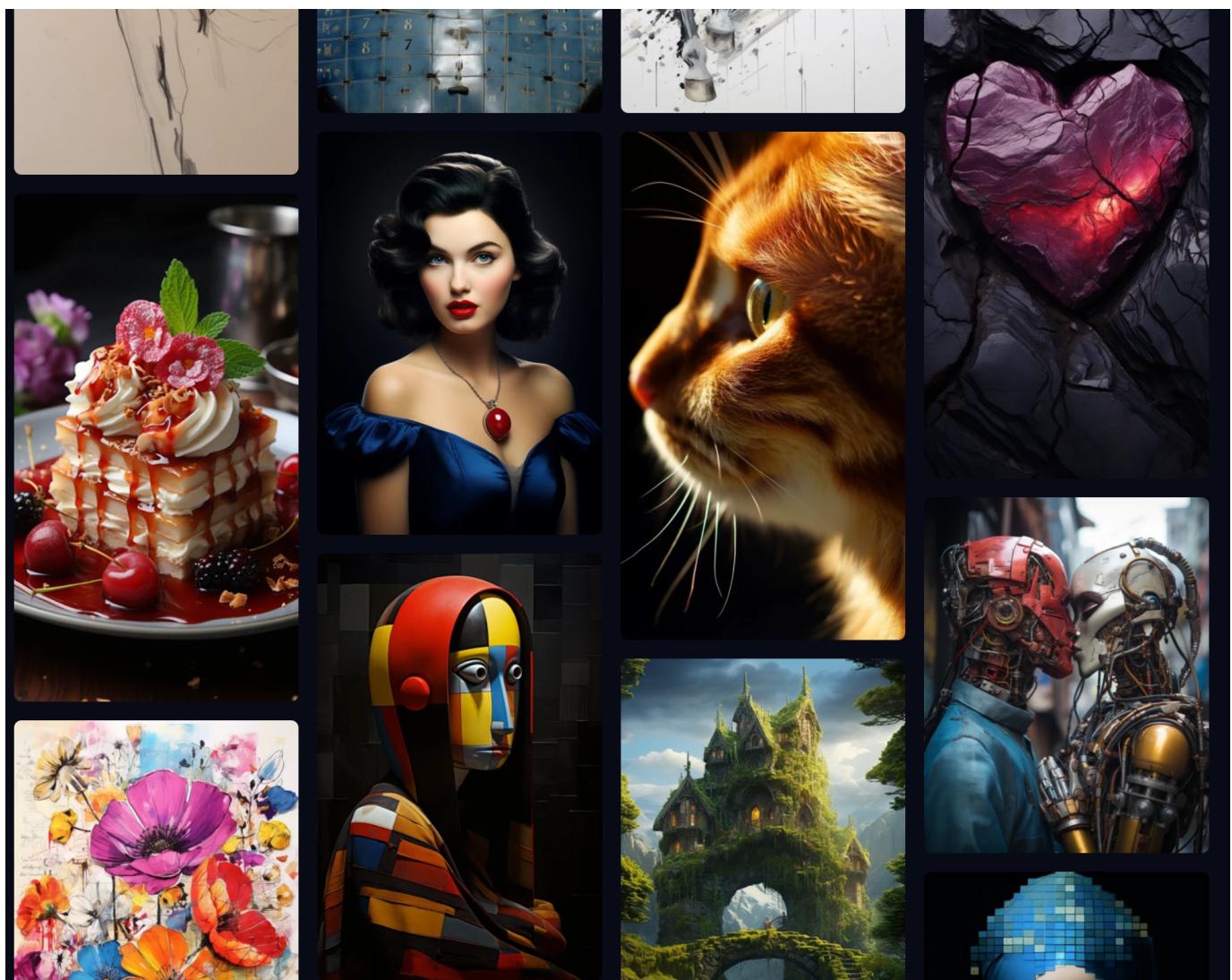
Procesos creativos

Uno de los grandes e inesperados avances de la computación de la década pasada, ha sido el de los modelos generativos: las redes [GAN](#) para el campo de la imagen y los modelos de generación de texto basados en Transformers.

- **Deep Dream:** En 2015 apareció DeepDream, un modelo de generación de imágenes creado por Google. El software Deep Dream fue desarrollado para el imageNet large scale visual recognition challenge (ILSVRC). Este era un desafío reto, propuesto a diferentes equipos de investigación, que consistió en crear un sistema de reconocimiento de objetos y su localización dentro de una misma imagen, aparte de su detección inmediata. En este Desafío se adjudicó a Google el primer premio en el año 2014, logrado gracias al uso del entrenamiento de redes neuronales. En junio de 2015 Google publicó la investigación, y tras esto hizo su código fuente abierto utilizado para generar las imágenes en un IPython notebook. Con esto se permitió que las imágenes de la red neuronal pudiesen ser creadas por cualquiera. Actualmente, se puede utilizar la aplicación de manera online en la web [DeepDreamGenerator](#). Básicamente, el algoritmo procesa la imagen dada identificando sus elementos, para utilizar una transferencia de estilos respetando la identidad esencial de la imagen original.
- **Gaugan:** GauGAN es una herramienta con la que se pueden crear paisajes falsos partiendo de un boceto. Este software de Nvidia, hace uso de una red de confrontación generativa (GAN), basado en una técnica denominada "normalización espacialmente adaptativa" que es capaz de generar imágenes realistas a partir de un determinado diseño semántico, controlado por el usuario con el uso de un programa de edición de imágenes, donde cada color actúa como representación de un tipo de objeto, material o ambiente. Se puede utilizar desde su interfaz web abierta.

- **DALL-E:** Una de las más recientes incorporaciones al ámbito de "cosas increíbles que la inteligencia artificial puede hacer ya" es el modelo de generación de imágenes de openAI. Se trata de una implementación multimodal de GPT3. El algoritmo interpreta una descripción escrita que se le proporciona a través de su interfaz, y genera la imagen correspondiente en base a lo que sus 12 mil millones de parámetros del modelo GPT3 han interpretado de la entrada de texto dada. En concreto, se utiliza un proceso llamado "diffusion" que parte de una imagen de ruido aleatoria y va alterando dicho esquema de puntos en función de que vaya reconociendo distintos patrones de objetos cuyas palabras clave se le han dado en la descripción.
- **MidJourney:** Midjourney es un laboratorio independiente de investigación y el nombre de un programa de inteligencia artificial con el cual sus usuarios pueden crear imágenes a partir de descripciones textuales, similar a Dall-e de OpenAI y al Stable Diffusion de código abierto.

La herramienta funcionó bajo versión de beta cerrada hasta que el 13 de julio de 2022 el laboratorio anunció el comienzo de una beta abierta. El equipo de Midjourney está dirigido por David Holz, cofundador de Leap Motion. Midjourney emplea un modelo de negocio *freemium*, con un nivel gratuito limitado y niveles de pago que ofrecen un acceso más rápido, mayor capacidad y funciones adicionales. Los usuarios pueden crear obras de arte con Midjourney dando órdenes a un bot alojado en Discord, ya sea enviando mensajes directos o invitando a dicho bot a un servidor de terceros.



Procesamiento del Lenguaje Natural (PLN)

El Procesamiento del Lenguaje Natural (PLN) es una rama de la Inteligencia Artificial que se enfoca en permitir a las máquinas entender y procesar el lenguaje humano en forma escrita o hablada. El PLN permite que las computadoras analicen, comprendan y generen texto de manera similar a como lo hacen los seres humanos. Esta tecnología ha sido fundamental en el desarrollo de asistentes virtuales, traducción automática, análisis de sentimientos y muchas otras aplicaciones útiles en el ámbito empresarial y cotidiano.

Tratar computacionalmente una lengua implica un proceso de modelización matemática. Los ordenadores solo entienden de bytes y dígitos y los informáticos codifican los programas empleando lenguajes de programación como C, Python o Java.

Los lingüistas computacionales se encargan de la tarea de "preparar" el modelo lingüístico para que los ingenieros informáticos lo implementen en un código eficiente y funcional.

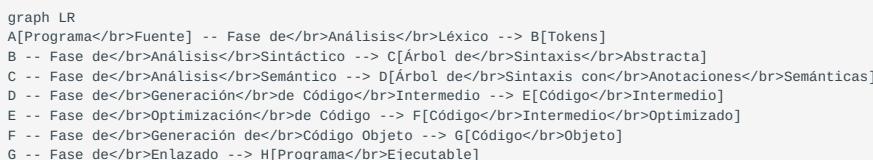
Éstos son algunos de los **componentes** del procesamiento del lenguaje natural. No todos los análisis que se describen se aplican en cualquier tarea de PLN, sino que depende del objetivo de la aplicación.

- **Análisis morfológico o léxico.** Consiste en el análisis interno de las palabras que forman oraciones para extraer lemas, rasgos flexivos, unidades léxica compuestas. Es esencial para la información básica: categoría sintáctica y significado léxico.
- **Análisis sintáctico.** Consiste en el análisis de la estructura de las oraciones de acuerdo con el modelo gramatical empleado (lógico o estadístico).
- **Análisis semántico.** Proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas.
- **Análisis pragmático.** Incorpora el análisis del contexto de uso a la interpretación final. Aquí se incluye el tratamiento del lenguaje figurado (metáfora e ironía) como el conocimiento del mundo específico necesario para entender un texto especializado.

Un análisis morfológico, sintáctico, semántico o pragmático se aplicará dependiendo del objetivo de la aplicación. Por ejemplo, un conversor de texto a voz no necesita el análisis semántico o pragmático. Pero un sistema conversacional requiere información muy detallada del contexto y del dominio temático.



No te recuerda a algo?



Estas son las fases de un Compilador

APLICACIONES DE PROCESAMIENTO DEL LENGUAJE NATURAL

Asistentes Virtuales y Chatbots

Los asistentes virtuales como Siri, Google Assistant y Alexa utilizan PLN para entender y responder a las consultas y comandos de voz de los usuarios. Los chatbots en aplicaciones de servicio al cliente y soporte técnico también emplean PLN para ofrecer respuestas automáticas y contextuales a las preguntas de los usuarios.

Esta generación actual de asistentes están habilitados para llevar a cabo tareas dentro del sistema que las aloja e, incluso, a través de webhooks, en otros sistemas que cuenten con las políticas de acceso correspondientes. De esta forma, los asistentes virtuales más avanzados pueden encargar una pizza, comprar online un producto entre varias sugerencias o incluso controlar la domótica de nuestra casa.

Generación de textos

El área del marketing y comunicación ha sido de los primeros que ha abrazado la inteligencia artificial para automatizar y mejorar muchos de sus procesos. Y entre las distintas tareas que puede llevar a cabo la IA, la generación de textos empezó a tener una aplicación comercial clara como herramienta para crear mensajes publicitarios, publicaciones de marketing de contenidos o incluso lemas de producto. Por eso, durante estos años han ido surgiendo una gran cantidad de servicios de este tipo. Siempre para generar breves fragmentos, y con una serie de requerimientos, como incluir las palabras clave.

Pero, recientemente, están surgiendo modelos mucho más generales y con una versatilidad mayor en el tipo de textos, el tema a tratar, el idioma, etc. Es el caso de los modelos BERT, GPT3 y Bloom. El primero, creado por Google en 2018, integrado en el algoritmo de búsqueda de Google y publicado con licencia de código abierto, no tiene una aplicación directa de generación de textos, pero tiene la misma arquitectura que los modelos que se están utilizando en ese campo: los Transformers.

Interpretación de textos

En el campo del análisis de lenguaje natural existen aplicaciones basadas en voz, como los asistentes virtuales que tenemos encima de la mesa, en el móvil o en el ordenador, o las aplicaciones basadas en texto. Ambas utilizan la misma base, y después, para añadir la habilitación oral, se utiliza un módulo de "Voz a Texto" y viceversa.

Las aplicaciones de PLN sirven para extraer información valiosa de los datos sin estructurar basados en textos y para acceder a la información extraída con el objetivo de generar una nueva comprensión de esos datos. Algunos ejemplos de aplicación serían:

- Traducción automática de idiomas.
- Chatbots.
- Opinión de los clientes: Se usa el análisis de entidades para identificar y etiquetar campos en documentos y canales. De esta forma, se pueden conocer mejor las opiniones de los clientes y obtener información valiosa sobre los productos y la experiencia de usuario.
- Comprender los recibos y las facturas: Extrae entidades para identificar las entradas más comunes de los recibos y facturas, como las fechas o los precios, y entiende la relación entre la solicitud y el pago.
- Análisis de documentos: Utiliza la extracción de entidades personalizada para identificar las entidades específicas de cada dominio en los documentos sin tener que invertir tiempo o dinero en análisis manuales.
- Clasificación de contenido general: Clasifica los documentos en función de las entidades más frecuentes, las entidades personalizadas de un dominio concreto o categorías generales disponibles (por ejemplo, deportes y entretenimiento).
- Análisis de tendencias: Agrega noticias con texto que permita a los profesionales del marketing extraer contenido relevante sobre sus marcas de noticias online, artículos y otras fuentes de datos.
- Sanidad: Mejora la documentación clínica, la investigación de minería de datos y los informes de registros automatizados para agilizar los ensayos clínicos.

El campo del procesamiento del lenguaje natural es considerado uno de los grandes retos de la inteligencia artificial ya que es una de las tareas más complicadas y desafiantes: ¿cómo comprender realmente el significado de un texto? ¿cómo intuir neologismos, ironías, chistes o poesía?

Análisis de Sentimientos

El PLN es utilizado para analizar el contenido de opiniones, comentarios y reseñas de usuarios en línea y determinar si expresan sentimientos positivos, negativos o neutrales. Esto es útil para medir la satisfacción del cliente, realizar estudios de mercado y realizar análisis de reputación de marca.

Ejemplo de Aplicación de Procesamiento del Lenguaje Natural: Análisis de Sentimientos

Un ejemplo práctico de aplicación de Procesamiento del Lenguaje Natural es el análisis de sentimientos en comentarios de productos en línea. En este caso:

- **Entrada:** La entrada es un conjunto de comentarios escritos por usuarios sobre un producto específico.
- **Procesamiento de Lenguaje Natural:** El PLN procesa el texto para tokenizarlo (dividirlo en palabras), eliminar palabras irrelevantes (stopwords) y realizar lematización o extracción de raíces para reducir las palabras a su forma base.
- **Análisis de Sentimientos:** Se utilizan técnicas de análisis de sentimientos para asignar un valor de sentimiento (positivo, negativo o neutral) a cada comentario en función de las palabras y frases utilizadas.
- **Salida:** Como resultado, se obtiene un resumen del sentimiento general de los usuarios hacia el producto, lo que permite a las empresas identificar puntos fuertes y áreas de mejora, así como tomar decisiones basadas en la retroalimentación del cliente.

Analítica avanzada

Los Modelos Predictivos son un grupo de técnicas que, mediante los campos del aprendizaje automático, la recolección de datos históricos, el Big Data y el reconocimiento de patrones, pretende dar una predicción de resultados futuros; con el objetivo de precisar la toma de decisiones mediante técnicas de análisis de datos. En los últimos años el área predictiva ha tomado gran protagonismo en los negocios, la medicina, los servicios financieros, las políticas gubernamentales, la publicidad, la mercadotecnia, las redes sociales y gran cantidad de campos de aplicación.

Se basa, principalmente en datos organizados tabularmente. Es decir, hablamos de datos estructurados y bases de datos relacionales en la mayoría de los casos. Se busca el patrón de comportamiento y la tendencia escondida en las relaciones entre diferentes variables de un sistema, y, a través de aprendizaje supervisado, con modelos de regresión y de clasificación, se obtienen predicciones que ayudan a la toma de decisiones en la organización.

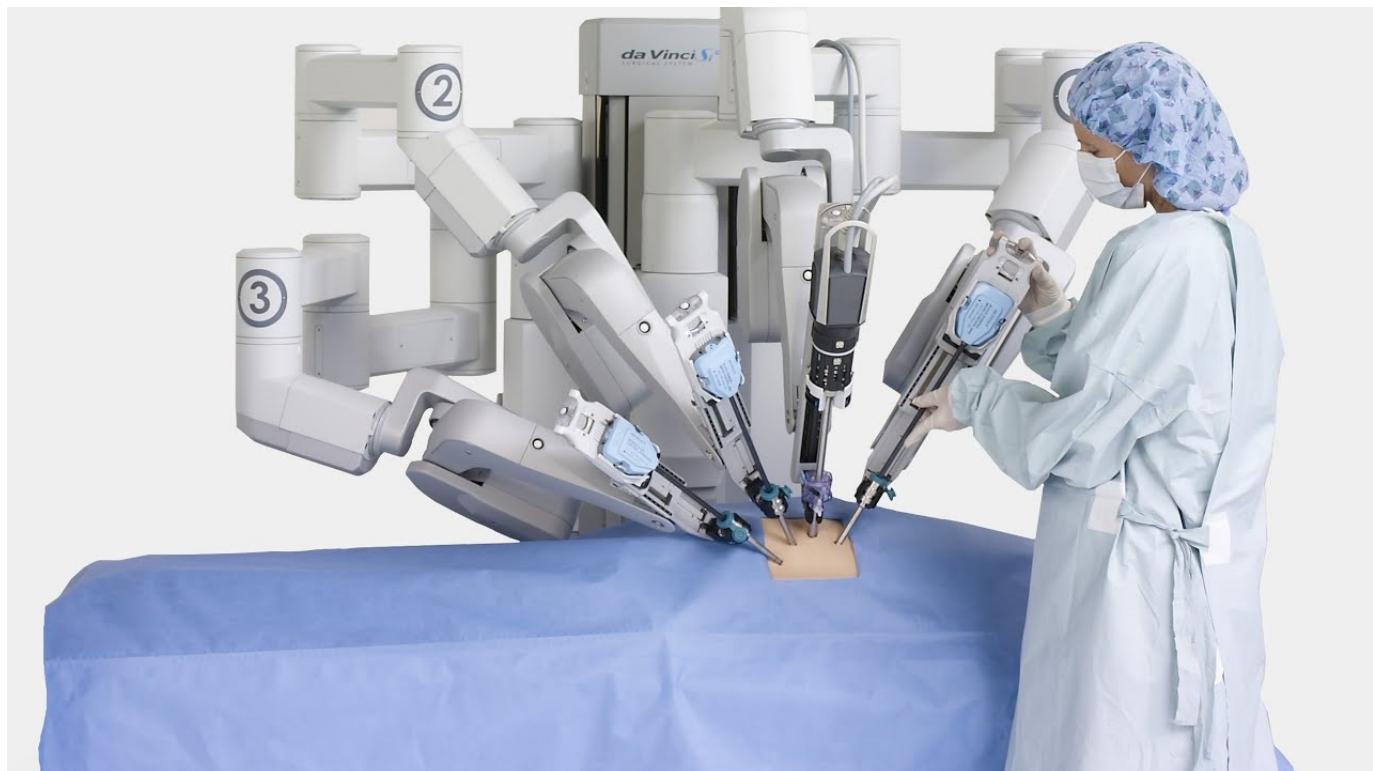
Cada vez van siendo más utilizados también los modelos de aprendizaje automático no supervisado, como el "clustering", que son el alma de sistemas de recomendación en plataformas de contenido online o comercio electrónico.

Los modelos predictivos tienen gran aplicabilidad en todos los sectores comerciales. Son capaces de resolver muchos problemas que antes eran irresolubles.

Robótica e Inteligencia Artificial

La integración de la IA en la robótica ha llevado a la creación de robots cada vez más inteligentes y autónomos, lo que ha revolucionado diversas áreas de aplicación.

Los sistemas robóticos actuales cubren una gran cantidad de campos de aplicación del entorno. En todos ellos, la inteligencia artificial mejora su desempeño y permite acometer nuevas tareas. Entre ellos, destacan muchos robots cuya principal herramienta basada en inteligencia artificial es el módulo de visión artificial, como es el caso de Davinci, el robot cirujano, o agro-bot, el robot que recoge fresas en su punto óptimo de madurez.



La inteligencia artificial tiene un impacto especialmente relevante en el sistema de control del robot.

APLICACIONES DE LA IA EN LA ROBÓTICA

Robots sociales.

Este tipo de robots tienen muy desarrollados los módulos sensoriales, es decir, el de reconocimiento de imagen, procesamiento de lenguaje natural, y su sistema de control es, básicamente, un asistente virtual que tiene ciertas opciones de movilidad y acciones remotas o conectadas, como encender la luz o hacer una llamada de emergencia.

Casas y ciudades inteligentes.

Estos sistemas robóticos cuentan con un elaborado sistema de sensores, y un hardware extendido por diversas localizaciones, lo que hace necesario contar, a menudo con microcontroladores que hagan parte del procesamiento de la información que captan y luego lo envíen al controlador principal. Por ejemplo, un sistema en el que tenemos cámaras que monitorizan las ventanas, en el propio microcontrolador de la cámara se puede hacer la tarea de reconocer la imagen de "ventana abierta", que la unidad principal recibirá junto a otros datos de interés como si está lloviendo o si hace frío, para accionar un actuador que haga saltar una alerta en el móvil de la persona propietaria o incluso que accione un motor para cerrarla.

Conducción autónoma.

La unidad de control de un vehículo autónomo es el paradigma de las técnicas más avanzadas en aprendizaje automático. Se trata de aprendizaje por refuerzo, y se entrena en simuladores virtuales hasta que el sistema tiene un comportamiento más o menos aceptable como para probarlo de forma segura en circuitos de pruebas reales.

Existen vehículos autónomos desde hace bastante tiempo, como es el caso de los drones, e incluso el sistema de control de navegación de los aviones cuenta con muchos automatismos, pero, probablemente, un coche autónomo, hoy por hoy, es el sistema más espectacular en tanto debe lidiar con muchos obstáculos y reglas de circulación.

Robots Colaborativos

Los robots colaborativos, también conocidos como cobots, son robots diseñados para trabajar de forma segura y eficiente junto a los seres humanos. Estos robots se utilizan en la industria para aumentar la productividad y mejorar la seguridad en la colaboración humano-robot. Por ejemplo:

- **Ensamblaje de Productos:** En líneas de ensamblaje de automóviles o electrónicos, los cobots pueden trabajar junto a los operadores humanos para realizar tareas repetitivas y pesadas, mejorando la eficiencia y reduciendo la fatiga de los trabajadores.
- **Embalaje y Logística:** En almacenes y centros de distribución, los cobots pueden colaborar con los trabajadores en la selección, embalaje y envío de productos, agilizando los procesos logísticos.

Robótica Médica

La robótica médica ha revolucionado la cirugía y la asistencia médica, permitiendo procedimientos más precisos y menos invasivos. Algunos ejemplos de aplicaciones de la robótica médica son:

- **Cirugía Asistida por Robot:** Los robots quirúrgicos, controlados por cirujanos, pueden realizar movimientos más precisos y estables durante procedimientos complejos, reduciendo el riesgo de errores y acelerando la recuperación del paciente.
- **Rehabilitación Robótica:** Los robots de rehabilitación se utilizan para asistir en la terapia de pacientes con discapacidades físicas, proporcionando movimientos controlados y repetitivos para mejorar la recuperación.
- **Cuidados de Pacientes:** Los robots asistenciales pueden ayudar a los pacientes en tareas diarias, como levantarse, caminar y tomar medicamentos, brindando una mayor autonomía y apoyo en el cuidado de la salud.

Robots Autónomos

Los robots autónomos son máquinas que pueden operar de manera independiente en entornos desconocidos o adversos sin intervención humana directa. Algunos ejemplos de aplicaciones de robots autónomos incluyen:

- **Exploración Espacial:** Los robots autónomos se utilizan en misiones espaciales para explorar planetas, asteroides y lunas, recopilando datos valiosos sin la necesidad de una comunicación constante con la Tierra.
- **Búsqueda y Rescate:** En situaciones de desastres naturales o emergencias, los robots autónomos pueden buscar y localizar supervivientes en áreas peligrosas o inaccesibles para los equipos de rescate humanos.

Ciencia de datos y Data Mining

La ciencia de datos es una de las disciplinas en las que la inteligencia artificial ha generado un mayor impacto, permitiendo detectar patrones y relaciones mediante métodos no supervisados, y llevar a cabo agrupaciones y heurísticos. Todo ello será visto con detalle en el módulo SAA, donde se encontrarán los algoritmos y aplicaciones más conocidos. En este campo se engloban también los heurísticos y los detectores de anomalías para planes de mantenimiento industrial.

Minería de datos o Data mining

No es raro ver cómo se usan indiferentemente los conceptos minería de datos y machine learning. Son conceptos "primos hermanos", pero no son lo mismo.

La principal diferencia radica en el objetivo que tiene cada una de las disciplinas. Mientras que la minería de datos descubre patrones anteriormente desconocidos, el Machine Learning se usa para reproducir patrones conocidos y hacer predicciones basadas en los patrones.

En pocas palabras se podría decir que la minería de datos tiene una función exploratoria mientras que el machine learning se focaliza en la predicción.

Ciberseguridad

A día de hoy, la IA juega un papel fundamental en el campo de la ciberseguridad. Algunos de los usos más destacados de la IA en ciberseguridad incluyen:

1. **Detección de amenazas avanzadas:** La IA se utiliza para analizar grandes volúmenes de datos y detectar patrones de comportamiento anómalos que puedan indicar actividades maliciosas. Los sistemas de detección de intrusiones basados en IA pueden identificar amenazas avanzadas y desconocidas que los enfoques tradicionales podrían pasar por alto.
2. **Prevención de ataques de phishing:** Los algoritmos de aprendizaje automático pueden analizar correos electrónicos y sitios web en busca de indicios de phishing y ayudar a bloquear o filtrar contenido malicioso antes de que llegue a los usuarios.
3. **Identificación de malware:** Los sistemas de IA pueden analizar el comportamiento de archivos y aplicaciones para detectar malware y ransomware. Además, la IA puede mejorar la identificación y clasificación de nuevas variantes de malware desconocido.
4. **Autenticación de usuarios:** La IA se utiliza en sistemas de autenticación biométrica y de reconocimiento facial para mejorar la seguridad en el acceso a dispositivos y servicios.

5. **Análisis de logs y eventos de seguridad:** Los sistemas de IA pueden analizar y correlacionar grandes cantidades de registros y eventos de seguridad para identificar patrones de actividad sospechosa y facilitar la respuesta a incidentes.
6. **Predicción y prevención de brechas de seguridad:** Mediante el análisis de datos históricos y la identificación de vulnerabilidades conocidas, la IA puede predecir posibles brechas de seguridad y ayudar a prevenir futuros ataques.
7. **Automatización de tareas de seguridad:** La IA se utiliza para automatizar tareas repetitivas en ciberseguridad, como la gestión de parches, el análisis de vulnerabilidades y la respuesta a incidentes, lo que permite a los profesionales de seguridad centrarse en tareas más complejas.
8. **Protección de redes y sistemas IoT:** La IA puede monitorear y proteger redes empresariales y dispositivos IoT (Internet de las cosas) para detectar y prevenir actividades maliciosas.

Ejemplos de soluciones reales en ciberseguridad basadas en IA incluyen:

- **Cylance:** Una plataforma de prevención de ataques basada en IA que utiliza algoritmos de aprendizaje automático para proteger contra malware y ransomware.
- **Darktrace:** Un sistema de detección de amenazas basado en IA que utiliza algoritmos de inteligencia artificial para identificar y responder a comportamientos anómalos en tiempo real.
- **IBM Watson for Cyber Security:** Una solución de seguridad cibernetica que utiliza IA para analizar grandes cantidades de datos y ayudar a identificar y responder a amenazas de manera más rápida y precisa.
- **BioCatch:** Una solución de autenticación biométrica que utiliza IA para analizar el comportamiento del usuario y detectar actividades fraudulentas.



Nuevas Formas de Interacción

Interfaces de Voz

Las interfaces de voz son una forma de interacción con sistemas de Inteligencia Artificial que permiten a los usuarios comunicarse mediante comandos de voz en lugar de texto o clics.

Estas interfaces han experimentado un crecimiento significativo en popularidad gracias a avances en el procesamiento del lenguaje natural y la tecnología de reconocimiento de voz. A continuación, se desarrollarán algunas aplicaciones y ejemplos de interfaces de voz.

ASISTENTES VIRTUALES

Los asistentes virtuales son aplicaciones de inteligencia artificial que permiten a los usuarios realizar tareas y obtener información a través de comandos de voz. Algunos ejemplos de asistentes virtuales populares incluyen:

- **Siri:** Desarrollado por Apple, Siri es un asistente virtual que se encuentra integrado en dispositivos como iPhones, iPads y Macs. Puede responder preguntas, enviar mensajes, configurar alarmas, hacer llamadas y más.
- **Google Assistant:** Desarrollado por Google, Google Assistant está disponible en dispositivos Android y en otros dispositivos como Google Home. Puede proporcionar información en tiempo real, realizar búsquedas en la web, establecer recordatorios y controlar dispositivos inteligentes del hogar.
- **Amazon Alexa:** Alexa es el asistente virtual de Amazon y está presente en dispositivos como el Amazon Echo. Permite realizar compras en línea, reproducir música, controlar luces y termostatos inteligentes, y realizar muchas otras tareas.

SISTEMAS DE NAVEGACIÓN

Las interfaces de voz también se utilizan en sistemas de navegación para proporcionar instrucciones de conducción en tiempo real. Algunos ejemplos incluyen:

- **Google Maps:** La popular aplicación de mapas y navegación utiliza el reconocimiento de voz para que los conductores reciban instrucciones mientras mantienen la vista en la carretera.
- **Sistemas de Navegación Integrados en Automóviles:** Muchos automóviles modernos están equipados con sistemas de navegación por voz que permiten a los conductores obtener direcciones y encontrar lugares de interés sin quitar las manos del volante.

APLICACIONES DE ACCESIBILIDAD

Las interfaces de voz también juegan un papel crucial en la accesibilidad tecnológica para personas con discapacidades visuales o motrices. Algunos ejemplos de aplicaciones de accesibilidad incluyen:

- **Lectores de Pantalla:** Estas aplicaciones utilizan la voz para leer en voz alta el contenido de la pantalla de un dispositivo, permitiendo que las personas con discapacidades visuales puedan interactuar con la tecnología.
- **Comandos de Voz para Controlar Dispositivos:** Las interfaces de voz permiten a personas con discapacidades motrices controlar dispositivos y realizar tareas sin la necesidad de utilizar las manos.

Interfaces Cerebro-Computadora (BCI)

Las Interfaces Cerebro-Computadora (Brain-Computer Interface BCI) son tecnologías avanzadas que permiten la comunicación directa entre el cerebro humano y dispositivos tecnológicos. A través del registro y análisis de señales cerebrales, estas interfaces posibilitan que las personas controlen dispositivos y sistemas mediante su actividad cerebral.

Algunas aplicaciones destacadas son:

ASISTENCIA MÉDICA

Las BCI han abierto nuevas posibilidades para asistir a personas con discapacidades motoras. Estas interfaces permiten a individuos con lesiones espinales, amputaciones u otras condiciones que afectan su capacidad de movimiento, controlar prótesis y dispositivos asistenciales mediante señales cerebrales. Por ejemplo, una persona con una extremidad amputada podría usar una prótesis controlada por BCI para realizar movimientos precisos y naturales, restaurando parte de su funcionalidad física.

NEUROFEEDBACK

El neurofeedback es una técnica terapéutica en la que se proporciona a los individuos información en tiempo real sobre su actividad cerebral. Las BCI se utilizan para capturar señales cerebrales y mostrar a los usuarios visualizaciones de sus patrones de actividad cerebral. Esto permite que las personas aprendan a autorregular su actividad cerebral, lo que puede ser beneficioso para tratar problemas de salud mental, como el estrés, la ansiedad y la depresión, y mejorar el rendimiento cognitivo en tareas específicas.

JUEGOS Y ENTRETENIMIENTO

Las BCI también se aplican en el campo de los juegos y el entretenimiento. Al utilizar señales cerebrales para controlar videojuegos y aplicaciones, se crea una experiencia de juego más inmersiva e interactiva. Los jugadores pueden controlar personajes y acciones dentro del juego mediante su actividad cerebral, lo que abre un mundo de posibilidades para nuevas mecánicas de juego y experiencias innovadoras.

Realidad Aumentada y Virtual

La Realidad Aumentada (AR) y la Realidad Virtual (VR) son tecnologías que mezclan el mundo digital con el mundo real o generan entornos completamente simulados para el usuario. Estas tecnologías ofrecen nuevas formas de interactuar con la IA y el mundo digital, proporcionando experiencias inmersivas y enriquecedoras.

Algunas aplicaciones destacadas son:

EDUCACIÓN Y FORMACIÓN

La AR y la VR se están convirtiendo en herramientas valiosas en el campo de la educación y la formación. Al simular entornos y escenarios de la vida real, estas tecnologías permiten a los estudiantes aprender de manera práctica y vivencial, lo que puede mejorar significativamente la retención de conocimientos y habilidades. Por ejemplo, los estudiantes de medicina pueden realizar simulaciones de cirugías en entornos de realidad virtual, lo que les proporciona una experiencia práctica sin riesgos para los pacientes.

DISEÑO Y VISUALIZACIÓN

Las AR y VR también ofrecen ventajas en el campo del diseño y la visualización. Los arquitectos, ingenieros y diseñadores pueden utilizar estas tecnologías para ver y modificar modelos 3D de edificios, productos o espacios. Esto permite una comprensión más clara y detallada de los proyectos, lo que puede conducir a un diseño más eficiente y una toma de decisiones más informada.

ENTRETENIMIENTO E INMERSIÓN

En el ámbito del entretenimiento, la AR y la VR proporcionan experiencias inmersivas que integran elementos del mundo real y virtual. Los videojuegos de realidad virtual permiten a los jugadores sumergirse por completo en mundos virtuales y participar en experiencias interactivas. Además, la AR se ha utilizado en aplicaciones de entretenimiento móvil, como juegos de realidad aumentada que interactúan con el entorno del usuario.



<https://sataraseguridad.com/2021/03/31/la-realidad-virtual-llega-al-entrenamiento-policial/>

Mapa conceptual

```

mindmap
root)"</br>[Inteligencia Artificial]</br>(
[</br>Fundamentos]
(Definición)
(Historia)
(Futuro)
(Sistemas<br>Inteligentes)
[Clasificaciones]
(Según tareas)
(Escuelas)
(Russell/Norvig)
(Hintze)
[Utilizacion<br>Modelos]
    
```

(Requisitos
de un SRP)
(Modelos de
sistemas de IA)

[Técnicas]
(Sistemas
expertos)
(Machine
Learning)
(Redes
Neuronales)
(Algoritmos
Genéticos)
(Lógica
Difusa)

[Aplicaciones]
(Visión por
Computadora)
(PLN)
(Analítica
avanzada)
(Robótica
e IA)
(Ciencia de datos
y Data Mining)
(Ciberseguridad)

[Interacción]
(Voz)
(Cerebro
Computadora)
(RA y RV)

14 de julio de 2025

2.2 Diapositivas de la UD01

-  Parte 1 de 5
-  Parte 2 de 5
-  Parte 3 de 5
-  Parte 4 de 5
-  Parte 5 de 5

⌚14 de julio de 2025

2.3 Actividades

Taller UD01_01: Preparar entorno para Java

Cada software y cada entorno de desarrollo tiene unas características y funcionalidades específicas. Esto también se verá reflejado en la instalación y configuración del software. Dependiendo de la plataforma, entorno o sistema operativo en el que se vaya a instalar el software, se utilizará un paquete de instalación u otro, y habrá que tener en cuenta unas opciones u otras en su configuración. A continuación se muestra cómo instalar una herramienta de desarrollo de software integrada, como Eclipse. Pero también podrás observar los procedimientos para instalar otras herramientas necesarias o recomendadas para trabajar con el lenguaje de programación JAVA, como Tomcat o la Máquina Virtual de Java. Debes tener en cuenta los siguientes conceptos:

- La JVM (Java Virtual Machine, máquina virtual de Java) es la encargada de interpretar el bytecode y generar el código máquina del ordenador (o dispositivo) en el que se ejecuta la aplicación. Esto quiere decir que necesitamos una JVM distinta para cada entorno.
- JRE (Java Runtime Environment) es un conjunto de utilidades Java que incluye la JVM, las bibliotecas y el conjunto de software necesario para ejecutar aplicaciones cliente Java, así como el conector para que los navegadores de Internet ejecuten applets.
- JDK (Java Development Kit) es el conjunto de herramientas para desarrolladores; contiene, entre otras cosas, el JRE y el conjunto de herramientas necesarias para compilar el código, empaquetarlo, generar documentación...

```
graph TD
    TD[graph TD]
    TD --> JDK[subgraph JDK]
    JDK --> JRE[subgraph JRE]
    JRE --> JVM[subgraph JVM]
    JVM --> end1[end]
    end1 --> end2[end]
```

El proceso de instalación consta de los siguientes pasos: 1. Descargue, instale y configure el JDK. 2. Descargue e instale un servidor web o de aplicaciones. 3. Descargue, instale y configure el IDE (Netbeans o Eclipse). 4. Configurar JDK con IDE. 5. Configure el servidor web o de aplicaciones con el IDE instalado. 6. Si es necesario, instalación de conectores. 7. Si es necesario, instale un nuevo software.

Descargue e instale el JDK

Podemos diferenciar entre:

- Java SE (Java Standard Edition): es la versión estándar de la plataforma, siendo esta plataforma la base para todos los entornos de desarrollo Java ya sea de aplicaciones cliente, de escritorio o web.
- Java EE (Java Enterprise Edition): esta es la versión más grande de Java y generalmente se utiliza para crear grandes aplicaciones cliente/servidor y para el desarrollo de servicios web.

En este curso se utilizarán las funcionalidades de Java SE. El archivo es diferente según el sistema operativo donde se tenga que instalar. Así:

- Para los sistemas operativos Windows y Mac OS hay un archivo instalable.
- Para los sistemas operativos GNU/Linux que admiten paquetes .rpm o .deb, también están disponibles paquetes de este tipo.
- Para el resto de sistemas operativos GNU/Linux existe un archivo comprimido (terminado en .tar.gz).

En los dos primeros casos, simplemente hay que seguir el procedimiento de instalación habitual del sistema operativo con el que estamos trabajando. En este último caso, sin embargo, hay que descomprimir el archivo y copiarlo en la carpeta donde se desea instalar. Normalmente, todos los usuarios tendrán permisos de lectura y ejecución en esta carpeta.

A partir de la versión 11 de JDK, Oracle distribuye el software con una licencia significativamente más restrictiva que las versiones anteriores. En particular, solo se puede utilizar para "desarrollar, probar, crear prototipos y demostrar sus aplicaciones". Cualquier uso "para fines comerciales, de producción o empresariales internos" distinto del mencionado anteriormente queda explícitamente excluido.

Si lo necesitas para alguno de estos usos no permitidos en la nueva licencia, además de las versiones anteriores del JDK, existen versiones de referencia de estas versiones licenciadas "GNU General Public License version 2, with the Classpath Exception", que permiten la mayoría de los usos habituales. Estas versiones están enlazadas a la misma página de descarga y también a la dirección jdk.java.net.

Una alternativa es utilizar <https://adoptium.net/> antes conocido como adoptOpenJDK, que ahora se ha integrado en la fundación Eclipse. Desde allí podemos descargar los binarios de la versión openJDK para nuestra plataforma sin restricciones. [Noticia completa] (<https://es.wikipedia.org/wiki/OpenJDK>).

En GNU/Linux podemos utilizar los comandos:

- `sudo apt install default-jdk` para instalar el jdk predeterminado.
- `java --version` para ver las versiones disponibles en nuestro sistema.
- `sudo update-alternatives --config java` para elegir cuál de las versiones instaladas queremos usar por defecto o incluso ver la ruta de las diferentes versiones que tenemos instaladas.

Configurar las variables de entorno "JAVA_HOME" y "PATH"

Una vez descargado e instalado el JDK, debes configurar algunas variables de entorno:

- La variable `JAVA_HOME`: indica la carpeta donde se ha instalado el JDK. No es obligatorio definirla, pero es muy cómodo hacerlo, ya que muchos programas buscan en ella la ubicación del JDK. Además, resulta muy fácil definir las dos variables siguientes.
- La variable `PATH`. Debe apuntar al directorio que contiene el ejecutable de la máquina virtual. Suele ser la subcarpeta `bin` del directorio donde hemos instalado el JDK.

Variable CLASSPATH Otra variable que tiene en cuenta el JDK es la variable `CLASSPATH`, que apunta a las carpetas donde se encuentran las librerías de la aplicación que se quiere ejecutar con el comando `java`. Es preferible, no obstante, indicar la ubicación de estas carpetas con la opción `-cp` del mismo comando `java`, ya que cada aplicación puede tener diferentes librerías y las variables de entorno afectan a todo el sistema. Establecer la variable `PATH` es esencial para que el sistema operativo encuentre los comandos JDK y pueda ejecutarlos.

IntelliJ

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) escrito en Java para desarrollar software informático escrito en Java, Kotlin, Groovy y otros lenguajes basados en JVM. Está desarrollado por JetBrains (antes conocido como IntelliJ) y está disponible como una edición comunitaria con licencia Apache 2 y en una edición comercial propietaria. Ambas se pueden utilizar para el desarrollo comercial.

Nuestra institución dispone de licencias para nuestros alumnos mientras tengáis correo electrónico @ieseduardoprimo.es.

INSTALACIÓN

Descargue desde <https://www.jetbrains.com/idea/> la versión de la herramienta toolbox correspondiente a su sistema operativo.

Siga las instrucciones para su sistema operativo desde <https://www.jetbrains.com/help/idea/installation-guide.html#toolbox>

Una vez instalada la caja de herramientas, puede elegir instalar todos los productos de JetBrains.

Una vez instalada la Idea (IDE) puedes crear una entrada de escritorio desde la pantalla inicial:

Create desktop icon

Y en la opción Administrar licencias debes seguir estas instrucciones: https://www.jetbrains.com/help/license_server/Activating_license.html

La dirección del servidor es: <https://ieseplm.flx.jetbrains.com/>

AJUSTES

Documentos para configurar su IDE: <https://www.jetbrains.com/help/idea/configuring-project-and-ide-settings.html>

MÓDULOS

Puedes agregar complementos siguiendo estas instrucciones:

<https://www.jetbrains.com/help/idea/managing-plugins.html>

USO BÁSICO ("¡HOLA MUNDO!")

Los documentos te ayudan con tu primer programa en Java: <https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html>

Mucha más información:

- Si vienes de Eclipse: <https://www.jetbrains.com/help/idea/migrating-from-eclipse-to-intellij-idea.html>
- Si estuvieras en NetBeans: <https://www.jetbrains.com/help/idea/netbeans.html>
- Si quieres aprender por tu cuenta: <https://www.jetbrains.com/help/idea/product-educational-tools.html>

Por qué debería elegir IntelliJ en lugar de VsCode para la codificación en Java

IDEA INTELLIJ:

Ventajas:

- Entorno integrado completo:** IntelliJ IDEA está diseñado específicamente para el desarrollo de Java y ofrece un conjunto completo de herramientas y características optimizadas para esta tarea.
- Análisis estático avanzado:** Proporciona un análisis de código en profundidad que detecta errores y problemas potenciales antes de la compilación.
- Depuración avanzada:** ofrece un potente conjunto de herramientas de depuración que ayudan a identificar y resolver problemas en el código.
- Refactorización guiada:** Proporciona herramientas para reorganizar y optimizar el código de forma segura, promoviendo buenas prácticas de programación.
- Compatibilidad con marcos y tecnologías Java:** Integración nativa con muchos marcos y tecnologías utilizados en el desarrollo Java, lo que facilita la creación de aplicaciones completas.
- Generación automática de código:** ayuda a los programadores a generar automáticamente fragmentos de código repetitivos, como captadores y definidores.
- Integración con herramientas de compilación:** facilita la integración con herramientas de compilación como Maven y Gradle.
- Sopporte para pruebas unitarias:** Ofrece integración con marcos de prueba como JUnit para el desarrollo basado en pruebas.
- Facilidad de configuración:** Proporciona asistentes guiados para configurar de manera eficiente proyectos Java.

Contras:

- Mayor consumo de recursos:** Debido a su naturaleza integral y rica en funciones, IntelliJ IDEA puede consumir más recursos del sistema en comparación con IDE más livianos.
- Curva de aprendizaje:** Dado que ofrece una amplia gama de funciones, los principiantes pueden tardar un tiempo en familiarizarse con todas las herramientas disponibles.

VISUAL STUDIO CODE (VSCODE):

Ventajas:

- Ligero y rápido:** VSCode es un editor de código liviano y rápido, lo que lo hace ideal para proyectos más pequeños o para aquellos que prefieren una experiencia más ágil.
- Amplia gama de extensiones:** Tiene una amplia comunidad que desarrolla extensiones para diversas tecnologías y lenguajes, incluido Java.
- Versatilidad:** Si bien no está diseñado específicamente para Java, se puede personalizar para que funcione con Java a través de extensiones.
- Integración de control de versiones:** ofrece integración nativa con sistemas de control de versiones como Git.
- Curva de aprendizaje rápida:** Debido a su enfoque más ligero, puede resultar más sencillo para los principiantes comenzar a trabajar con él.

Contras:

- Funcionalidad limitada de Java:** Aunque existen extensiones de Java, VSCode no ofrece el mismo conjunto completo de herramientas optimizadas para Java que IntelliJ IDEA.
- Ánalisis menos profundo:** Las capacidades de análisis estático y corrección de código podrían no ser tan avanzadas como las de IntelliJ IDEA.
- Depuración limitada:** si bien ofrece depuración, es posible que no sea tan avanzada o completa como la de IntelliJ IDEA.
- Configuración manual del proyecto:** La configuración de proyectos Java puede requerir más pasos y configuración manual en comparación con IntelliJ IDEA.

Tarea

Debes entregar un documento *.pdf explicando:

Una captura de pantalla en la que se vea el resultado del comando:

- `java java --version`

Y también capturas de pantalla donde se pueda ver que editas el fichero fuente (`HolaMundo.java`), lo compilas y lo ejecutas dentro del IDE IntelliJ (explica los pasos que has seguido)

⌚14 de julio de 2025

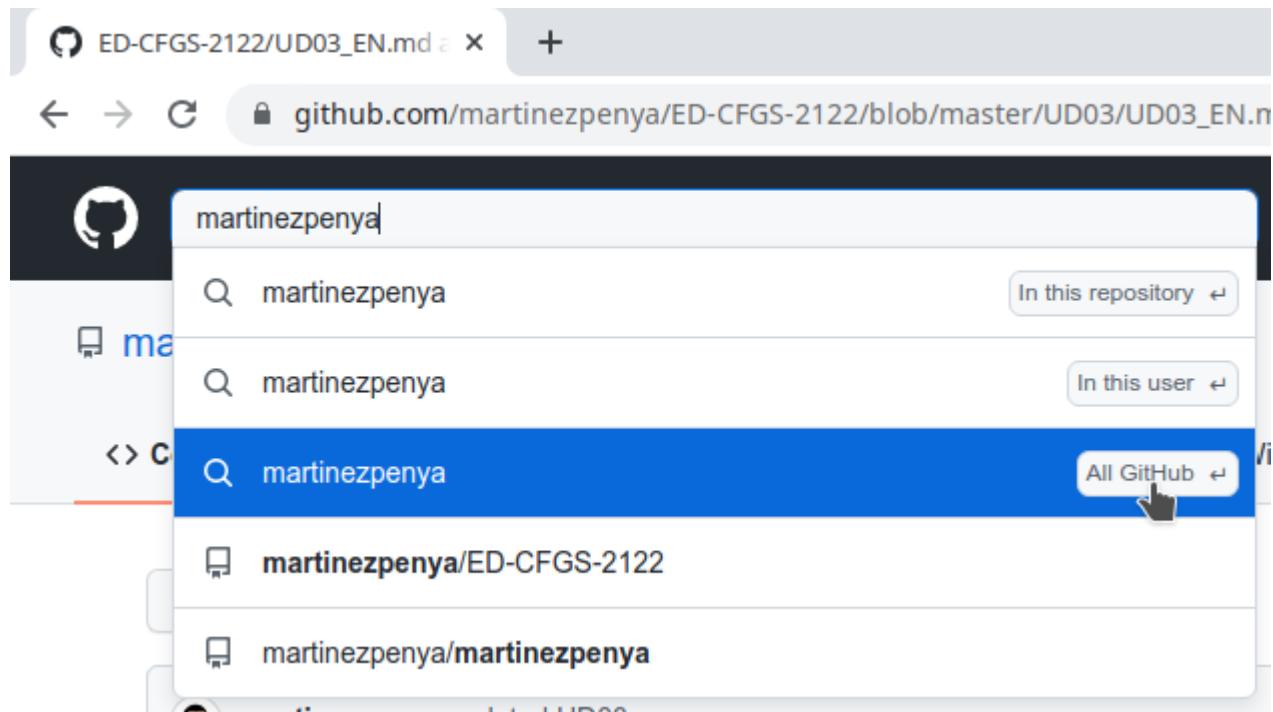
Taller UD01_02: vscode integrado en GitHub

Instrucciones

LOCALIZAR REPOSITORIO.

Primero localizamos el repositorio con el que queremos colaborar (Por favor, busca uno del año actual, la imagen es solo un ejemplo):

1. Buscamos al usuario (En todo GitHub):



1. Elegimos la pestaña `Repositorios`:

martinezpenya / README.md

Hi there 🙋 I'm David Martinez Peña and I...

Hola 🙋 Soy David Martinez Peña y e...

Hola 🙋 Soc David Martinez Peña i es...

Popular repositories

MyHAConfiguration
My HomeAssistant configuration

1. Elegimos el que nos interesa modificar, en nuestro caso PRG-CFGS-2324 :

Find a repository...

PRG-CFGS-2122 Public
Java GNU General Public License v3.0 Updated 4 hours ago

ED-CFGS-2122 Public
Java Updated 22 days ago

ABRIR vscode INTEGRADO.

Github cuenta con un editor online muy potente basado en vscode .

Una vez visualizamos el código del repositorio en cuestión:

martinezpenya / ED-CFGS-2122 Public

Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code

Commit	Message	Date
	martinezpenya updated UD03	7201523 22 days ago
	UD01 minor updates	4 months ago
	UD02 UD02 Updates	last month
	UD03 updated UD03	22 days ago
	UD04 rename to lowercase UD04_anexo_ES.*	29 days ago
	_code/DevelopmentEnvironments UD04 and _code folder	29 days ago
	.gitignore UD04 and _code folder	29 days ago
	README.md added README.md	4 months ago

Para abrir el editor solo debemos pulsar la tecla "." (punto) de nuestro teclado:

github.dev/martinezpenya/ED-CFGS-2122

EXPLORER [Preview] README.md

- ED-CFGS-2122 [GITHUB]
 - > _code
 - > UD01
 - > UD02
 - > UD03
 - > UD04
 - ≡ .gitignore
 - ⓘ README.md

Apuntes de teoría y ejercicios para la asignatura de Entornos de desarrollo (en inglés) de 1º de DAW del curso 21-22. IES Mestre Ramón Esteve (Catadau)

GitHub master Layout: es

Visualizaremos la estructura de carpetas y archivos en un editor vscode integrado en el navegador Web.

MODIFICAR UN ARCHIVO.

Una vez detectada la errata dentro del código **markdown** que es muy fácil de interpretar (a poco que le dediquéis unos minutos) podemos modificar el archivo en cuestión, y a su lado aparecerá una **M** porqué el archivo está modificado.

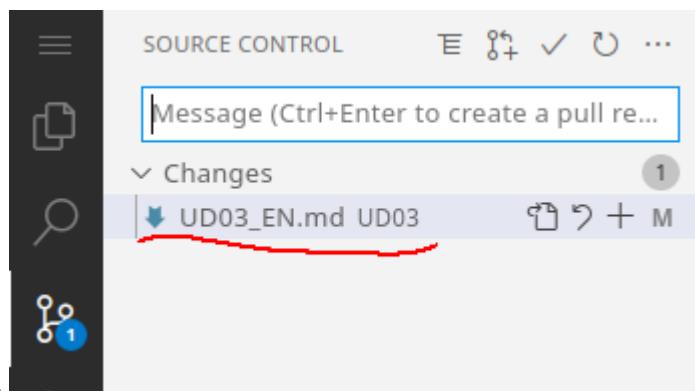
The screenshot shows the VSCode interface with the GitHub repository 'ED-CFGS-2122' selected in the Explorer sidebar. A red circle highlights the 'git' icon next to the repository name. The main editor area displays the file 'UD03_EN.md' with code comments and a preview of the document content.

GIT INTEGRADO

VSCode lleva integrado un gestor de GIT, el tercer ícono de la barra lateral:

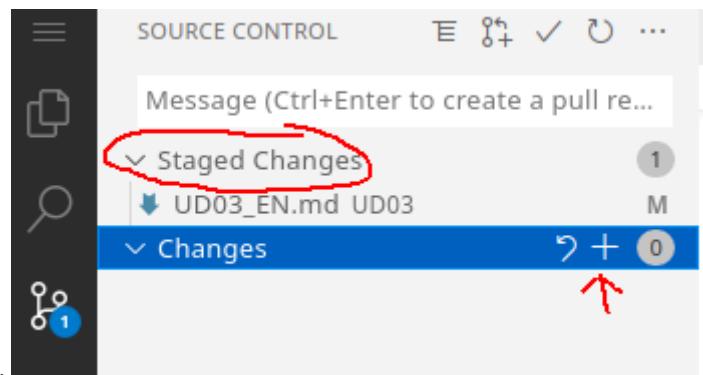


1.



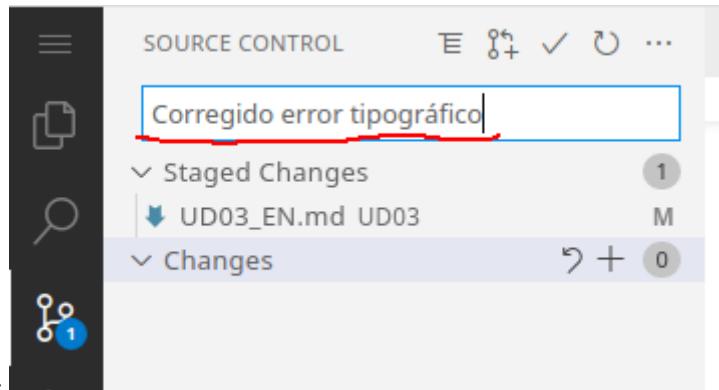
Verificamos los archivos cambiados:

2.



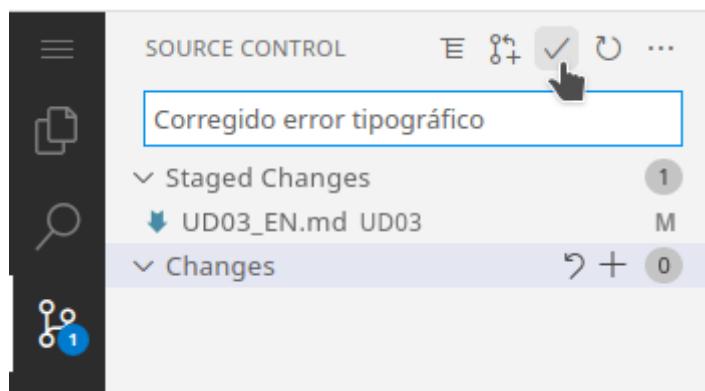
Los pasamos al área "staged" con el símbolo "+":

3.

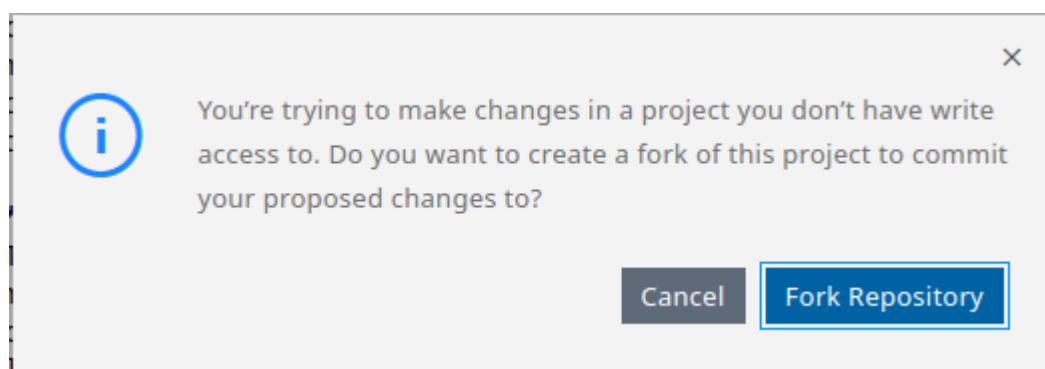


Añadimos el comentario del commit:

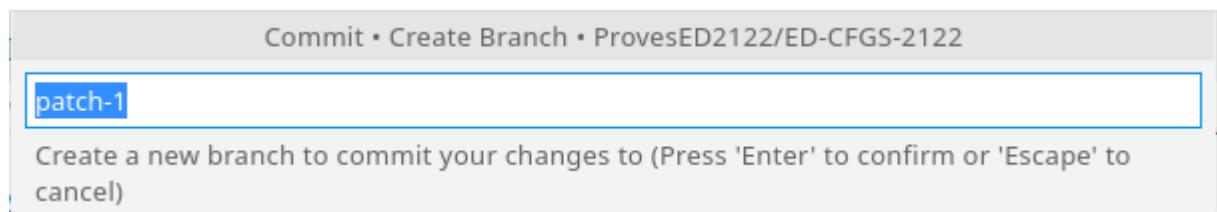
4.



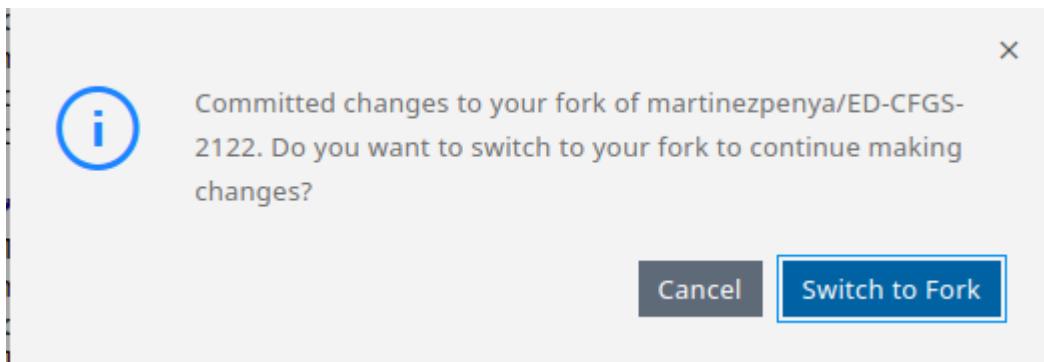
Realizamos el commit:

En realidad podemos hacer el fork antes o después, aquí tienes un pequeño [vídeo](#) que explica que es un fork.

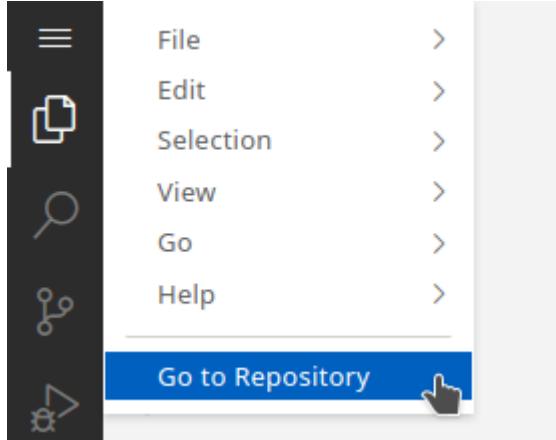
A continuación nos pide el nombre de la rama que se creará y que luego podremos solicitar se incluya en el proyecto original:



Ahora nos pregunta si ya hemos creado un fork, queremos cambiar el repositorio, y trabajar sobre nuestro fork en lugar de sobre el proyecto original, pulsamos sobre [Switch to Fork]:



Ahora ya podemos salir del editor VSCode pulsando sobre el botón de las tres líneas horizontales y elegir la opción "Go to Repository":



PULL REQUEST

Una vez volvemos a nuestro repositorio (nuestro fork), detectará que hay cambios respecto al repositorio original y nos propone que realicemos un pull request (una petición al usuario propietario del repositorio original para que incluya nuestra modificación).

Una vez pulsado el botón [Compare & pull request] nos aparece la siguiente pantalla:

[Open a pull request](#)

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

 ✓ **Able to merge.** These branches can be automatically merged.

 Corregido error tipográfico

Write Preview H B I ↵ ↶ ↷ ↸ ↹ ↻ ↺

Modificación realizada por David Martinez (profe)

Attach files by dragging & dropping, selecting or pasting them.

Allow edits by maintainers ?

Create pull request ▾

Debemos asegurarnos de que la modificación se puede agregar al repositorio original "Able to merge", y que indicamos en los comentarios nuestro nombre completo para que el profesor nos identifique. Fíjate que el nombre del pull request es el nombre del commit que hicimos desde VSCode.

Ahora debemos pulsar el botón [Create pull request].

Tarea

Sigue los pasos de este taller guiado para sugerir una modificación de cualquiera de los archivos de los repositorios del profesor [martinezpenya](#) (Preferiblemente del curso actual). Adjunta a la tarea de AULES un [.pdf](#) con la **captura** de pantalla similar a esta donde se vé que has solicitado el **pull request** y que estás esperando a que se integre en el repositorio original. Además, **explica** que significan cada uno de los **5 apartados** señalados en la captura:

Corregido error tipográfico #1

[Open](#) ProvesED2122 wants to merge 1 commit into [martinezpenya:master](#) from [ProvesED2122:patch-1](#)

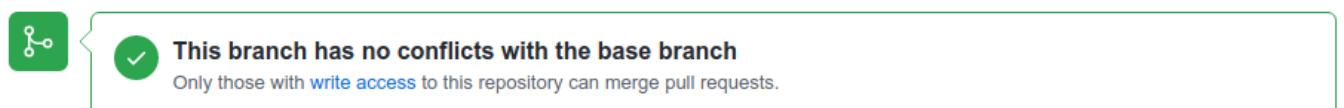
[Conversation](#) 0 [Commits](#) 1 [Checks](#) 0 [Files changed](#) 1

 ProvesED2122 commented 13 minutes ago

Modificación realizada por David Martínez (profe)

[Corregido error tipográfico](#) 6d7008f

Add more commits by pushing to the `patch-1` branch on [ProvesED2122/ED-CFGS-2122](#).



⌚14 de julio de 2025



Markdown nace como herramienta de **conversión de texto plano a HTML**. Fue creada en 2004 por John Gruber, y se distribuye de manera gratuita bajo una [licencia BSD](#).

Markdown es un maravilloso **lenguaje** para escribir documentos de una manera **sencilla de escribir, y que en todo momento mantenga un diseño legible** que contengan elementos como *secciones, párrafos, listas, vínculos e imágenes, etc.* Pandoc <http://pandoc.org> ha extendido enormemente la [sintaxis original de Markdown](#) y ha añadido unas pequeñas nuevas características tales como notas al pie de página, citas y tablas. Lo más importante que hace Pandoc es hacer posible la generación de documentos en una amplia variedad de formatos desde Markdown, HTML, LaTeX/PDF, MSWord y Slides.

Este método te permitirá añadir formatos tales como **negritas, cursivas o enlaces**, utilizando texto plano, lo que permitirá hacer de tu escritura algo más simple y eficiente al evitar distracciones.

Con Markdown **no vas a reemplazar todo**, sino cubrir las funcionalidades más comunes que se requieren para escribir un documento relativamente complicado.

Para qué sirve Markdown

Markdown será perfecto para ti sobre todo **si publicas de manera constante en Internet**, donde el lenguaje HTML está más que presente: WordPress, Squarespace, Jekyll...

Pero no estoy hablando solo de [blogs](#) o páginas web. **Servicios** como Trello o **foros** como Stackoverflow también soportan este lenguaje, y con el paso del tiempo encontrarás aún más lugares que lo utilicen.

Además, Markdown está cada vez más extendido en el **mundo “offline”**. Nada te impedirá utilizar este lenguaje para **tomar notas y apuntes** de tus clases o reuniones en una determinada **aplicación** (incluso podrías **escribir un libro con él**, ya que puedes exportar fácilmente el resultado final a un formato ePub).

Gracias a la simplicidad de su sintaxis podrás utilizarlo siempre que necesites escribir y dar formato rápidamente, sobre todo si quieras hacerlo desde dispositivos móviles.

Por qué utilizar Markdown

VENTAJAS

- **Markdown para todo.** Para crear apuntes, documentos, notas, sitios web, libros, documentación técnica, etc. de forma off-line.
- **Markdown transportable.** Este tipo de formato siempre será **compatible con todas las plataformas** que utilices, así que utilizar Markdown es una manera de mantener todo tu contenido siempre accesible desde cualquier dispositivo (smartphones, ordenadores de escritorio, tablets...), ya que en cualquiera de ellas siempre encontrarás **las aplicaciones adecuadas** para leer y editar este tipo de contenido.
- Ideal para escribir un libro, pues permite la exportación fácil en ePub, PDF...

Si en el futuro Microsoft Word desapareciese perderías acceso a todo el contenido que has creado durante años utilizando dicho procesador. Así que lo más inteligente para evitar eso es **generar tu contenido de la manera más sencilla posible**: utilizando texto plano.

DESVENTAJAS

- No tiene muchas funcionalidades (esto es lo que lo hace muy compatible).

Editores para Markdown

OFF-LINE

- **Typora**
- MarkdownPad
- HarooPad
- Markdown Monster
- ...

ONLINE

- Dillinger
- GitHub
- ...

Párrafos y saltos de línea

Si queremos generar un nuevo párrafo en Markdown simplemente separa el texto mediante una línea en blanco (**pulsando dos veces intro**).

Al igual que sucede con HTML, **Markdown no soporta dobles líneas en blanco**, así que si intentas generarlas estas se convertirán en una sola al procesarse.

Para realizar un salto de línea y empezar **una frase en una línea siguiente dentro del mismo párrafo**, tendrás que pulsar **dos veces la barra espaciadora antes de pulsar una vez intro**.

Por ejemplo si quisieses escribir un poema quedaría tal que así:

«*La tierra estaba seca, No había ríos ni fuentes. Y brotó de tus ojos.*

Donde cada verso tiene **dos espacios en blanco al final**.

Encabezados

Las **# almohadillas** son uno de los métodos utilizados en Markdown para crear encabezados. Debes usarlos añadiendo **uno por cada nivel**.

Es decir,

```
1 # Encabezado 1
2 ## Encabezado 2
3 ### Encabezado 3
4 #### Encabezado 4
5 ##### Encabezado 5
6 ##### Encabezado 6
```

Se corresponde con:

También puedes cerrar los encabezados con el mismo número de almohadillas, por ejemplo escribiendo **### Encabezado 3 ###**. Pero la única finalidad de esto es un **motivo estético**.

Texto básico

Un párrafo no requiere sintaxis especial.

Para aplicar **negrita** al texto, se escribe entre dos asteriscos.

Para aplicar *cursiva* al texto, se escribe entre un solo asterisco.

Para tachar el texto, se escribirá dos virgulillas antes y dos después de éste.

```
1 Este texto es en **negrita**.
2 Este texto es en *italica*.
3 Este texto está ~~tachado~~.
4 Este texto es en ambos ***negrita e italica***.
```

Se corresponde a:

Este texto es en ***negrita***.

Este texto es en *italica*.

Este texto está ~~tachado~~.

Este texto es en ambos ***negrita e italica***.

En Markdown no podemos subrayar el texto. Sin embargo, podremos añadir la etiqueta de html underline \u.

```
1 Este texto está <u>subrayado</u>
```

Este texto está subrayado

Para **ignorar los caracteres** de formato de Markdown, ponga _ antes del carácter:

Citas

Las citas se generan utilizando el carácter *mayor que* > al comienzo del bloque de texto.

```
1 > No hay que ir para atrás ni para darse impulso. — Lao Tsé.
```

No hay que ir para atrás ni para darse impulso. — Lao Tsé.

Si la cita en cuestión se compone de **varios párrafos**, deberás añadir el mismo símbolo > al comienzo de cada uno de ellos.

Listas

LISTAS ORDENADAS

Para crear **listas numeradas**, empieza una línea con **1.** or **1)**.

No debes mezclar los formatos dentro de la misma lista. No es necesario especificar los números. GitHub lo hace por tí.

```

1  1. ítem 1 de la lista.
2  1. Siguiente ítem de la lista.
3  1. Siguiente ítem, el tercero, de la lista.

```

Se corresponde con:

1. Ítem 1 de la lista.
2. Siguiente ítem de la lista.
3. Siguiente ítem, el tercero, de la lista.

LISTAS NO ORDENADAS

Para crear listas no numeradas, o de viñetas, empieza una línea con * , - o + , pero no mezcles los formatos dentro de la misma lista. (No mezclar formatos de viñetas, como * y + por ejemplo, dentro del mismo documento).

```

1  * ítem 1 de la lista.
2  * Siguiente ítem de la lista.
3  * Siguiente ítem, el tercero, de la lista.

```

Se corresponde con:

- Ítem 1 de la lista.
- Siguiente ítem de la lista.
- Siguiente ítem, el tercero, de la lista.

También podremos combinar ambos tipos de listas. Como por ejemplo:

1. element de llista 2
2. element de llista 2.2
 - element de llista 2.2.1
 - element de llista 2.2.2

LISTAS DE TAREAS

Para crear listas de tareas basta con que empiece la línea con - [] , si queremos que no esté el check marcado, y - [x] , si queremos que esté el check marcado.

```

1  - [x] regar plantas.
2  - [ ] realizar ejercicios de programación.

```

Se corresponde con:

- regar plantas.
- realizar ejercicios de programación.

Tablas

Las tablas no forman parte de la especificación principal de Markdown, pero Adobe, en cierta forma, las admite.

Para generar una tabla utiliza la barra vertical | para generar filas y columnas.

Si insertamos guiones --- dentro de una celda crearemos el encabezado de la tabla.

```

1  | encabezado1 | encabezado2 | encabezado3 |
2  |---|---|---|
3  | celda 1.1 | celda 1.2 | celda 1.3 |
4  | celda 2.1 | celda 2.2 | celda 2.3 |

```

Quedaría:

encabezado1	encabezado2	encabezado3
celda 1.1	celda 1.2	celda 1.3
celda 2.1	celda 2.2	celda 2.3

Si queremos una **celda con más de una línea** de texto podremos insertar \n (o Shift+Intro) al final de ésta.

Enlaces

Para generar enlace en Markdown se debe poner un código con dos partes:

- [texto del enlace], que es el texto que se va a mostrar,
- Y después (nombrefichero.md), que es la URL o el nombre de archivo al que se va a vincular.

```
1 [link text](file-name.md)
```

Un ejemplo:

[enlace a web del centro](https://iesmre.com)

La visualización del ejemplo anterior:

[enlace a web del centro](https://iesmre.com)

Imágenes

Para insertar una imagen se debe poner un código con dos partes:

- ! [texto alternativo], que es el texto que se va a mostrar si la imagen no pudiera visualizarse,
- Seguido de (nombrefichero.extension), que es el archivo imagen (con su dirección).

```
1 [texto alternativo](file-name.md)
```

Un ejemplo:

[logo markdown](/assets/mardown_logo.png)

La visualización de la imagen anterior:



Código de bloque

Uno de los puntos más útiles de Markdown a la hora de crear un documento con texto específico de informática es que admite la colocación de bloques de código tanto en línea como en un bloque "delimitado" independiente entre frases.

Para ello utilizaremos:

- Dos comillas invertidas `` si queremos escribir código dentro de la misma línea de texto del párrafo.
- Si queremos crear un bloque de código multilínea, con un lenguaje específico, pondremos ``` seguido del nombre del lenguaje del bloque .

Unos ejemplos:

- En la misma línea:

...estamos escribiendo un párrafo ``insertar el bloque y seguimos escribiendo...

- Un bloque de código:

```javascript y escribimos el código.

```
1 function holamundo(){
2 console.log ("hola mundo web");
3 }
```

## Línea horizontal

Para crear una línea horizontal, de separación de contenido por ejemplo, se añaden tres guiones: ---

Visualización:

---

### Insertar emojis

Para insertar emojis basta con utilizar `:` seguido del nombre del emoji y cerrar con otro `:`.

Podemos observar, que en algunos editores markdown, al escribir, por ejemplo, `:a` nos muestra todos los emojiis con la inicial **a**.

Por ejemplo: `: star :`

Visualización: 

### Crear diagrama de flujo

Cuando queremos crear documentos con elementos gráficos como diagramas de flujo, debemos generar una especie de *código* para construirlos.

- Por eso, comenzaremos introduciendo la línea de inicio: ````flow`
- Es conveniente asignar un nombre (por ejemplo: st, op, cond, e...) a cada elemento que conforma el diagrama; así, después podremos unir todos estos.
- Forma de inicio: `st=>start: Nombre`
- Forma de fin: `e=>end: Nombre`
- Rectángulo: `op=>operation: texto de nombre`
- Condición: `cond=>condition: texto de la condición (Si o No?)`
- Subrutina: `sub1=>subroutine: nombre subarea`
- EntradaSalida: `io1=>inputoutput: nombre elemento entrada/salida`
- Líneas: `st->op->cond`
- Caminos de condiciones: `cond(yes)**->**e` y `cond(no)->op`
- Línea de cierre: `````

Ejemplo:

```

1 ```flow
2 st=>start: Usuario
3 e=>end: Acceso
4 op=>operation: Operacion de usuario
5 cond=>condition: Si o No?
6 st->op->cond
7 cond(yes)->e
8 cond(no)->op
9 ```


```

Visualización:

```

1 st=>start: Usuario
2 op=>operation: Operacion de usuario
3 cond=>condition: Si o No?
4 e=>end: Acceso
5
6 st->op->cond
7 cond(yes)->e
8 cond(no)->op


```

Intenta realizar un diagrama para "programar" un almuerzo. En él, deberás dar los **buenos días**, indicar que **es hora del descanso**, y preguntar si **alguién quiere almorzar**. Si no hay nadie que quiera almorzar contigo, debes **ir a otro grupo de amigos** y volver a indicar que **es hora del descanso**. Si alguien sí quiere almorzar **escribe en la pizarra que os vais a almorzar y sal al patio**.

### Crear secuencias

/media/DADES/OneDriveGVA/DOCENCIA/24-25/media/DADES/OneDriveGVA/DOCENCIA/24-25En la secuenciación podemos observar que es bastante parecido a la creación de diagramas; pero la primera línea (crear un bloque de código) no será **flow** sino **sequence**.

```

1 ````sequence
2 Ana->Mundo: Hola Mundo
3 Note right of Mundo: Mundo está pensando\nla respuesta
4 Mundo-->Ana: Cómo estás?
5 Ana->>Mundo: Estoy bien gracias!
6 ```

```

Visualización:

```

1 Ana->Mundo: Hola Mundo
2 Note right of Mundo: Mundo está pensando\nla respuesta
3 Mundo-->Ana: Cómo estás?
4 Ana->>Mundo: Estoy bien gracias!

```

## Crear índice

Para crear el índice a partir de los encabezados creados debemos insertar `[TOC]`.

## Tarea

Como tarea, se propone:

- Crear un documento markdown en tu editor markdown favorito (por ejemplo Typora o VSCode) que documente información acerca de tí mismo.
- En dicho documento crear título, índice.
- Añadir 4 encabezados principales (y otros encabezados secundarios dentro de éstos) en el que hables por ejemplo de: *Tus datos, Currículum, Aficiones y Otros datos de interés*. No hace falta que indiques información personal relevante. (O te la puedes inventar)
- Se valorará la inclusión de distintos elementos como: negrita-cursiva-subrayado, listas ordenadas-desordenadas-tareas, enlaces, imágenes, citas, código, etc.
- Si te atreves con ello, crea un diagrama de flujo en el que indiques los pasos que realizas un sábado por la mañana.
- Exporta el documento a pdf.

**Subir a la plataforma **AULES** un documento Markdown (.md) y otro documento PDF (.pdf) que sea la exportación del primero.**

⌚14 de julio de 2025

## 2.4 Practicas

---

### Robocode TankRoyale

#### Preparación del entorno

- Al menos java 11 (Yo estoy usando Java 21 y la versión 0.24.4 de Tank Royale)

```
1 java -version
```

- Descargar la última versión desde releases: <https://github.com/robocode-dev/tank-royale/releases>
- Ejecutar el jar descargado:

```
1 java -jar robocode-tankroyale-gui-x.y.z.jar
```

- Descarga y descomprime los Bots de ejemplo: <https://github.com/robocode-dev/tank-royale/releases>
- Configura la gui para acceder a la carpeta de robots: Config -> Bot Root Directories (del menú)
- [opcional] Añadir sonidos al gui. Descarga y descomprime la carpeta sounds.zip de: <https://github.com/robocode-dev/sounds/releases>

```
1 [directorio padre]
2 |__ robocode-tankroyale-gui-x.y.z.jar
3 |__ sounds/ <- carpeta de sonidos
4 | |__ bots_collision.wav
5 | ...
6 | |__ wall_collision.wav
```

- Necesitaras IntelliJ para poner todo en funcionamiento y para programar tu Bot.

Juega un poco por tu cuenta con el entorno GUI, explora las opciones, tipos de batallas, crea alguna ronda de las mismas, inicializa Bots, añádelos, juega con la velocidad de juego, etc.

#### Robocode con Maven

Desde hace "relativamente" poco, robocode tankroyale está disponible como artefacto del repositorio de maven, si estas familiarizado con el uso de maven puede simplificar bastante el trabajo y actualizaciones de dependencias, te dejo el enlace por si quieres investigar por tu cuenta, ya que escapa del objetivo de esta asignatura: <https://central.sonatype.com/search?q=g:dev.robocode.tankroyale&smo=true>

#### Mi primer Bot

##### PROYECTO INTELLIJ

Para acelerar el proceso, he preparado un proyecto en IntelliJ con toda la arquitectura básica que necesitará tu Bot. Descarga y descomprime esta [carpeta](#) o [esta para maven](#) y abre el proyecto con el IDE IntelliJ.

También necesitaras la libreria (API) de Robocode Tank Royale que puedes descargar desde aquí: <https://github.com/robocode-dev/tank-royale/releases>. Descarga el archivo: `robocode-tankroyale-bot-api-x.y.z.jar`

La estructura debería quedar de la siguiente manera:

```
1 [directorio padre]
2 |__ lib
3 | |__ robocode-tankroyale-gui-x.y.z.jar
4 |__ IABDBot <- Proyecto de IntelliJ
5 | |__ src
6 | ...
7 | |__ IABDBot.iml
```

Asegúrate de entender el funcionamiento del Bot IABDBot, tiene comentarios donde se explican partes importante del Bot y que daremos por conocidas en los siguientes puntos.

Es muy importante que entiendas la diferencia entre movimientos bloqueantes y simultáneos, así como las condiciones y los eventos disponibles.

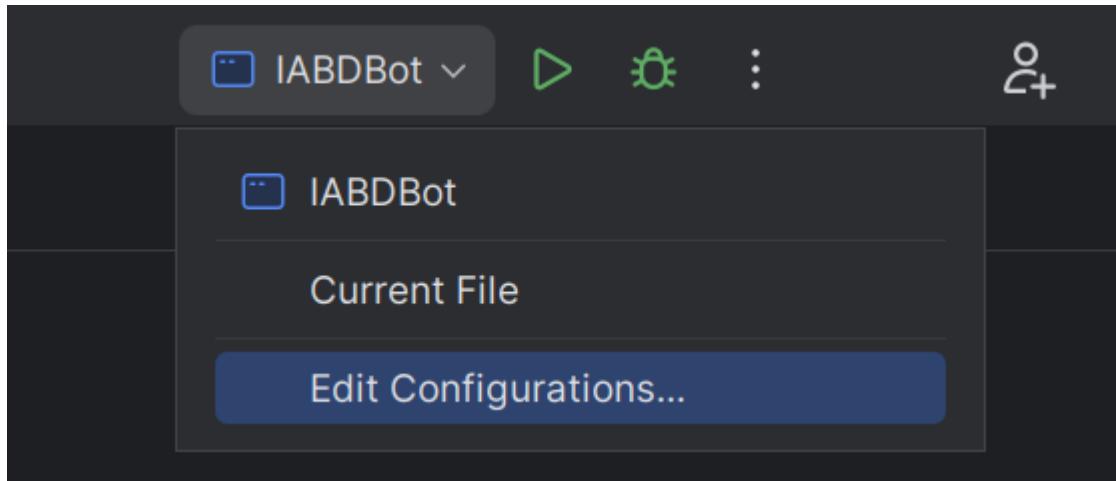
#### CONFIGURACIÓN DEL SERVIDOR (GUI) Y EL PROYECTO INTELLIJ PARA EJECUTAR EN LOCAL O REMOTO

El servidor permite conexiones en remoto/local a través de un password que se genera automáticamente la primera vez que lanzas la GUI. Este password lo puedes encontrar/modificar en el archivo `server.properties`, en el ejemplo siguiente la clave de acceso es **CEIABDEPM2024**

```
1 bots-secrets=CEIABDEPM2024
2 [...]
```

Otra información que necesitarás será la IP del servidor al que quieras conectar, normalmente será `localhost` (tu host local), y cuando sea necesario hacer pruebas el profesor te facilitará su IP.

Ahora solo queda configurar nuestro proyecto de IntelliJ para configurar estas variables en el momento de ejecutar el Bot. Accede a la configuración de las ejecuciones en la parte superior derecha (una vez abierta la clase `IABDBot.java`):



Puedes ver que en el apartado `Environment Variables` tienes el siguiente valor:

```
1 SERVER_SECRET=CEIABDEPM2024;SERVER_URL=ws://localhost:7654
```

Como puedes imaginar **SERVER\_SECRET** hace referencia a la clave del servidor, y **SERVER\_URL** a la dirección del servidor y es donde deberás reemplazar (según el caso) localhost por la IP que te indique el profesor. Así podrás ejecutar tu Bot directamente desde IntelliJ y aparecerá en el Servidor para poder añadirlo a las batallas.

Ojo, el servidor debe estar inicializado antes de lanzar el bot de lo contrario obtendrás un error similar a este:

```
1 Exception in thread "main" dev.robocode.tankroyale.botapi.BotException: Could not create web socket for URL: ws://localhost:7654
2 at dev.robocode.tankroyale.botapi.internal.BaseBotInternals.connect(BaseBotInternals.java:268)
3 at dev.robocode.tankroyale.botapi.internal.BaseBotInternals.start(BaseBotInternals.java:254)
4 at dev.robocode.tankroyale.botapi.BaseBot.start(BaseBot.java:114)
5 at IABDBot.main(IABDBot.java:11)
6
7 Process finished with exit code 1
```

Si el Bot encuentra el servidor y la contraseña es correcta debería aparecer esto al inicializarlo en IntelliJ:

```
1 Connected to: ws://localhost:7654
```

Prueba por tu cuenta mezclando en las batallas Bots de ejemplo, con tu Bot o incluso el de algún compañero/a.

#### ¿Cómo mejorar mi Bot?

Dividiremos todo lo que debemos conocer en 4 apartados:

##### CONOCIENDO EL CAMPO DE BATALLA

Sigue atentamente la documentación de RCTR sobre la **anatomía** de los Bots

Puntos importantes:

- Si el radar no se mueve, no detecta
- Inicialmente el robot, el cañón y el radar se mueven conjuntamente (pero se puede cambiar)
- El Bot se simplifica a un cuadrado de 36x36 unidades.
- Para las colisiones (con Bots o paredes) se simula como un círculo de 18 unidades de radio.

A continuación revisa las **coordenadas y ángulos**

Puntos importantes:

- Este apartado es totalmente diferente al de Robocode Original.

Estudia también las **físicas**

Puntos importantes:

- Se frena más rápido que se acelera.
- Cuanto más rápido vas, más lento giras.
- El cañón gira máximo 20º por turno.
- El radar gira máximo 45º por turno.
- La potencia de tiro influye en el daño provocado y los puntos conseguidos, pero también en el calentamiento del cañón.
- Inicialmente el cañón está caliente (3 unidades), al principio has de esperar a que se enfrie para disparar.
- El atropello da más puntos que los disparos (pero te resta energía)
- Si chocas con una pared, pierdes puntos.
- La energía que te sobra en una ronda no da puntos (modo kamikaze con el último robot?)

Por último te en cuenta las **puntuaciones**

Puntos importantes:

- Consigues puntos si:
  - tu disparo golpea a otro Bot (sino, te resta)
  - eres el que mata al Bot
  - cada vez que otro Bot muere, pero tu sigues vivo
  - eres el último robot vivo
  - atropellas a otro Bot
  - si matas por atropello a otro Bot
  - El último Bot que quede vivo no tiene porqué ser el ganador.

Eventos propios

Definimos una condición, y cuando esta se cumple se dispara un evento:

```

1 // Esta condición se dispara cuando el bot completa su giro
2 public static class TurnCompleteCondition extends Condition {
3
4 private final IBot bot;
5
6 public TurnCompleteCondition(IBot bot) {
7 this.bot = bot;
8 }
9
10 @Override
11 public boolean test() {
12 // El método test() es el que debemos reescribir para definir el resultado de nuestra condición.
13 return bot.getTurnRemaining() == 0;
14 }
15 }
```

Podriamos usar esta condición en un fragmento similar a este:

```
1 waitFor(new TurnCompleteCondition());
```

Podriamos usar funciones Lambda de la siguiente manera:

```
1 waitFor(new Condition(() -> getTurnRemaining() == 0));
```

El resultado de los dos fragmentos seria el mismo.

#### PERSONALIZAR MI BOT

Podemos establecer colores para el cuerpo, torreta de cañón, el radar, el arco de escaneo y de la bala:

```
1 Color kaki = Color.fromString("#febe56");
2 setBodyColor(kaki);
3 setTurretColor(kaki);
4 setRadarColor(kaki);
5 setScanColor(kaki);
6 setBulletColor(kaki);
```

#### TÉCNICAS DE ESCANEO (RADAR)

Sentidos de nuestro Bot:

Sentido del **tacto**, tu Bot sabe cuando:

- golpea una pared (`onHitWall`),
- es alcanzado por un disparo (`onHitByBullet`),
- es alcanzado por otro Bot (`onHitBot`).

Sentido de la **vista**, tu Bot sabe cuándo ha visto otro robot, pero sólo si lo escanea (`onScannedBot`)

**Otros** sentidos, tu robot también sabe cuándo ha muerto (`onDeath`), cuándo ha muerto otro robot (`onRobotDeath`).

Además también es consciente de sus balas y sabe cuando una bala ha sido disparada (`onBulletFired`) ha alcanzado a un oponente (`onBulletHit`), cuando una bala golpea una pared (`onBulletWall`) o cuando una bala golpea a otra bala (`onBulletHitBullet`).

Configurar correctamente tu Bot con:

```
1 setAdjustRadarForBodyTurn(true); //true if the radar must adjust/compensate for the body's turn;
2 //false if the radar must turn with the body turning (default).
3 setAdjustGunForBodyTurn(true); //true if the gun must adjust/compensate for the bot's turn;
4 //false if the gun must turn with the bot's turn.
5 setAdjustRadarForGunTurn(true); //true if the radar must adjust/compensate for the gun's turn;
6 //false if the radar must turn with the gun's turn.
```

Que el radar siempre de vueltas:

```
1 setTurnRadarLeft(Double.POSITIVE_INFINITY); //degrees - is the amount of degrees to turn left. If negative, the radar will turn right.
```

Revisa el API (`rescan()` i `setRescan()`)

Fijar el radar en un enemigo (one on one radar):

- Escaneo con multiplicador

```
1 public void onScannedBot(ScannedBotEvent e) {
2 double factor=1.9;
3 setTurnRadarLeft(factor * radarBearingTo(e.getX(), e.getY()));
4 }
```

Segun el factor:

- 1.0 - Bloqueo de radar delgado. Debe llamar a `scan()` para evitar perder el bloqueo. Que Dios te ayude si alguna vez te saltas un turno.
- 1.9 - El arco del radar comienza amplio y se estrecha lentamente tanto como sea posible mientras se mantiene en el objetivo.
- 2.0 : el arco del radar recorre un ángulo fijo. El ángulo exacto elegido depende de las posiciones del enemigo y del radar cuando se detecta al enemigo por primera vez. El ángulo se incrementará si es necesario para mantener el bloqueo.
- Arco de radar Ancho

```

1 public void onScannedBot(ScannedBotEvent e) {
2 //set the wide to add to the scan.
3 double wide=36.0;
4 // Absolute angle towards target
5 double angleToEnemy = radarBearingTo(e.getX(), e.getY());
6 // Distance we want to scan from middle of enemy to either side
7 // The 36.0 is how many units from the center of the enemy robot it scans.
8 double extraTurn = Math.min(Math.toDegrees(Math.atan(36.0 / distanceTo(e.getX(), e.getY()))), getMaxRadarTurnRate());
9
10 // Adjust the radar turn so it goes that much further in the direction it is going to turn
11 // Basically if we were going to turn it left, turn it even more left, if right, turn more right.
12 // This allows us to overshoot our enemy so that we get a good sweep that will not slip.
13 if (angleToEnemy < 0)
14 angleToEnemy -= extraTurn;
15 else
16 angleToEnemy += extraTurn;
17
18 //Turn the radar
19 setTurnRadarLeft(angleToEnemy);
20 }

```

- Radar Oscilante (variable global que sepa la dirección y nos ayude a cambiarla de vez en cuando)
- Escaneo más inteligente (seguir al cercano? según la situación?) ...
- Nos interesa mantener una lista de enemigos? e.getScannedBotId() ? y por ejemplo fijar el radar en el más débil?...

#### TÉCNICAS DE DESPLAZAMIENTO (MOVIMIENTO)

##### Pensamiento lateral

Si has jugado baloncesto antes, sabes que si quieres defender a alguien que sostiene el balón, debes maximizar tu movimiento lateral enfrentándote siempre a él (de frente). Lo mismo ocurre con tu robot.

Para conseguir esta posición debemos hacer algo similar a esto:

```
1 turnLeft(bearingTo(e.getX(), e.getY()) + 90);
```

que siempre colocará tu robot perpendicular (90 grados) a tu enemigo.

##### ¿Hacia adelante o hacia atrás?

Cuando te enfrentas a un oponente, la idea de "adelante" y "atrás" (del primer Bot) se vuelven algo obsoletas. Probablemente estés pensando más en términos de "atacar a la izquierda" o "atacar a la derecha". Para realizar un seguimiento de la dirección del movimiento, simplemente declare una variable como hicimos para oscilar el radar.

```
1 class MyRobot extends Bot {
2 private byte moveDirection = 1;
```

luego, cuando quieras mover tu robot, simplemente puedes decir:

```
1 setForward(100 * moveDirection);
```

Puedes cambiar de dirección cambiando el valor de moveDirection de 1 a -1 así:

```
1 moveDirection *= -1;
```

##### Cambiar de dirección

El enfoque más intuitivo para cambiar de dirección es simplemente cambiar la dirección del movimiento cada vez que golpeas una pared o golpeas a otro robot de esta manera:

```
1 public void onHitWall(HitWallEvent e) { moveDirection *= -1; }
2 public void onHitBot(HitBotEvent e) { moveDirection *= -1; }
```

Sin embargo, descubrirás que si haces eso, terminarás presionando obstinadamente contra un robot que te embiste desde un costado (como un perro en celo). Esto se debe a que se llama a `onHitRobot()` tantas veces que moveDirection sigue cambiando y nunca te alejas.

Un mejor enfoque es simplemente probar para ver si su robot se ha detenido. Si es así, probablemente significa que has golpeado algo y querrás cambiar de dirección. Puedes hacerlo con el código:

```

1 if (getSpeed() == 0)
2 moveDirection *= -1;

```

Ponlo en tu método `doMove()` (o en cualquier otro lugar donde estés manejando el movimiento) y podrás manejar todos los eventos de impacto con las paredes u otros Bots.

¿Bailamos?

## Dando vueltas (Círculos)

Puedes rodear a tu enemigo simplemente usando las técnicas anteriores:

```

1 public class IABDBot extends Bot {
2 double moveDirection=1;
3 private double ultimoEnemigoVisto;
4 ...
5 @Override
6 public void run() {
7 while (isRunning()) {
8 doMove();
9 go();
10 }
11 }
12 public void onScannedBot(ScannedBotEvent e) {
13 ...
14 ultimoEnemigoVisto=bearingTo(e.getX(), e.getY());
15 ...
16 }
17 public void doMove() {
18 // switch directions if we've stopped
19 if (getSpeed() == 0)
20 moveDirection *= -1;
21 // circle our enemy
22 setTurnLeft(ultimoEnemigoVisto+90);
23 setForward(1000 * moveDirection);
24 }
25 }

```

Objetivo: rodea a tu enemigo usando el código de movimiento anterior, como un tiburón rodeando a su presa en el agua.

## Evita Ametrallamiento

Un problema que puedes notar con el anterior tipo de movimiento es que es presa fácil para los objetivos predictivos porque sus movimientos son tan... predecibles.

Para evadir las balas de manera más efectiva, debes moverte de lado a lado o "atacar". Una buena forma de hacerlo es cambiar de dirección después de un cierto número de "tics", así:

```

1 public void doMove() {
2 // switch directions if we've stopped
3 if (getSpeed() == 0)
4 moveDirection *= -1;
5 // circle our enemy
6 setTurnLeft(ultimoEnemigoVisto+90);
7 if (getTurnNumber() % 20 == 0) {
8 moveDirection *= -1;
9 setForward(1000 * moveDirection);
10 }
11 }

```

Mira como se balancea hacia adelante y hacia atrás usando el código de movimiento anterior. Observa lo bien que esquiva las balas. Pero ojo, porque puede afectar a tu precisión de disparo.

## Cada vez más cerca

Notarás que tanto el primero como este último tipo de movimientos tienen otro problema: se atascan fácilmente en las esquinas y terminan golpeándose contra las paredes. Un problema adicional es que si su enemigo está lejos, disparan mucho pero no aciertan mucho.

Para hacer que tu robot se acerque a tu enemigo, simplemente modifica el código para que se gire ligeramente hacia su enemigo, así:

```

1 setTurnLeft(lastScannedBearing + (90 - (15 * moveDirection)));

```

15 es un factor en grados que puedes modificar y ajustar. Prueba!

Modificando el primero conseguimos una variación que utiliza el código anterior para lanzarse en espiral hacia su enemigo.

Mientras que con el segundo que utiliza el código anterior para acercarse cada vez más. También es bastante bueno para evadir balas.

Fíjate que ninguno de los Bots anteriores queda atrapado en una esquina por mucho tiempo.

#### Evitando las paredes

Un problema con todos los Bots anteriores es que golpean mucho las paredes y golpearlas agota tu energía. Una mejor estrategia sería detenerse antes de chocar contra las paredes. ¿Pero cómo?

### Agregar un evento personalizado

Lo primero que debemos hacer es decidir qué tan cerca permitiremos que nuestro robot llegue a las paredes:

```
1 public class WallAvoider extends Bot {
2 ...
3 private int wallMargin = 60;
```

A continuación, agregamos un evento personalizado que se activará cuando se cumpla una determinada condición dentro del método `run()`:

```
1 public void run() {
2 ...
3 // No te acerques mucho a las paredes
4 addCustomEvent(new Condition("TooCloseToWalls") {
5 public boolean test() {
6 // El método test() es el que debemos reescribir para definir el resultado de nuestra condición.
7 return (
8 // cerca de la pared izquierda
9 getX() <= wallMargin ||
10 // cerca de la pared derecha
11 getX() >= getArenaWidth() - wallMargin ||
12 // cerca de la pared inferior
13 getY() <= wallMargin ||
14 // cerca de la pared superior
15 getY() >= getArenaHeight() - wallMargin)
16);
17 });
18 });
19 ...
20 }
```

Ten en cuenta que estamos creando una clase interna anónima con esta llamada. Necesitamos hacer override del método `test()` para devolver un valor booleano cuando ocurra nuestro evento personalizado.

### Gestionar el evento personalizado

Lo siguiente que debemos hacer es manejar el evento, que se puede hacer así:

```
1 public void onCustomEvent(CustomEvent e) {
2 if (e.getCondition().getName().equals("TooCloseToWalls")) {
3 // switch directions and move away
4 moveDirection *= -1;
5 forward(100 * moveDirection);
6 }
7 }
```

Sin embargo, el problema con ese enfoque es que este evento podría dispararse una y otra vez, provocando que cambiamos rápidamente de un lado a otro, sin alejarnos nunca. O si ya aparecemos cerca de la pared quedamos atrapados.

Para evitar este "sacudida de la muerte" deberíamos tener una variable que indique que estamos manejando el evento. Podemos declarar otro así:

```
1 public class WallAvoider extends Bot {
2 ...
3 private int tooCloseToWall = 0;
```

Luego maneje el evento de manera un poco más inteligente:

```

1 public void onCustomEvent(CustomEvent e) {
2 if (e.getCondition().getName().equals("TooCloseToWalls")) {
3 if (tooCloseToWall <= 0) {
4 // Si no estábamos ya cerca de las paredes, ahora lo estamos
5 tooCloseToWall += wallMargin;
6 setMaxSpeed(0); // Para!!!
7 }
8 }
9 }

```

## Manejo de los dos modos

Hay dos últimos problemas que debemos resolver. En primer lugar, tenemos un método `doMove()` donde colocamos todo nuestro código de movimiento normal. Si estamos tratando de alejarnos de una pared, no queremos que se llame a nuestro código de movimiento normal, creando (una vez más) la "sacudida de la muerte". En segundo lugar, queremos volver eventualmente al movimiento "normal", por lo que eventualmente deberíamos tener el "tiempo de espera" de la variable `tooCloseToWall`.

Podemos resolver ambos problemas con la siguiente implementación de `doMove()`:

```

1 public void doMove() {
2 ...
3
4 // if we're close to the wall, eventually, we'll move away
5 if (tooCloseToWall > 0) tooCloseToWall--;
6
7 // switch directions if we've stopped
8 if (getSpeed() == 0) {
9 setMaxSpeed(8);
10 moveDirection *= -1;
11 setForward(1000 * moveDirection);
12 }
13 }

```

Con todo el código anterior evitamos chocar contra las paredes. Observa cómo se desliza suavemente hacia los lados pero nunca (bueno, rara vez) los golpea.

### Bot multimodo

Además de los colores que elijas, la mayor parte de la personalidad de tu robot está en su código de movimiento. Por otra parte, situaciones diferentes requieren tácticas diferentes. Usando el ejemplo de evitar paredes, es posible que deseas codificar tu bot para que cambie los "modos" según ciertos criterios. Podrías pensar en algo como esto:

```

1 public class MultiModeBot extends Bot {
2 private final static int MODO_RONDAR=0;
3 private final static int MODO_ESQUIVAR=1;
4 private final static int MODO_ASESINO=2;
5 private int modoRobot=MODO_RONDAR;
6 ...
7 }
8
9 public void onBotDeath(BotDeathEvent e) {
10 ...
11 if (getEnemyCount() > 10) {
12 // Un gran número de enemigos sugiere movimientos fluidos
13 modoRobot=MODO_RONDAR;
14 } else if (getEnemyCount() > 1) {
15 // esquivar es la mejor táctica para grupos pequeños
16 modoRobot=MODO_ESQUIVAR;
17 } else if (getEnemyCount() == 1) {
18 // Si solo queda un robot, persigue
19 modoRobot=MODO_ASESINO;
20 }
21 ...
22 }
23
24 public void doMove() {
25 switch (modoRobot){
26 case MODO_RONDAR:
27 //ronda
28 break;
29 case MODO_ESQUIVAR:
30 //esquiva
31 break;
32 case MODO_ASESINO:
33 //asesina
34 break;
35 }
36 ...
37 }

```

Los detalles se dejan (como siempre) como ejercicio para el alumnado.

## TÉCNICAS DE ATAQUE (DISPARO)

### Técnicas básicas

- Separa el movimiento del cañón del movimiento del Bot (`setAdjustGunForBodyTurn`)
- Técnica de apuntado básica: `setTurnGunLeft` o `setTurnGunRight` junto con `gunBearingTo`

### Fórmula de cálculo de potencia de fuego

Otro aspecto importante al disparar es calcular la potencia de fuego de tu bala. La documentación del método `fire()` explica que puedes disparar una bala en el rango de `0.1` a `3.0`. Como probablemente ya habrás concluido, es una buena idea disparar balas de baja potencia cuando tu enemigo está lejos y balas de alta resistencia cuando está cerca.

```

1 public void smartFire(double distance){
2 if ((distance > 200) || (getEnergy() < 15)){
3 fire(1);
4 } else if (distance > 50){
5 fire(2);
6 } else {
7 fire(3);
8 }
9 }
```

Podrías usar una serie de declaraciones `if-else-if-else` para determinar la potencia de fuego, en función de si el enemigo está a 50 unidades, a 200, etc (como en el fragmento anterior). Pero tales construcciones son demasiado rígidas. Después de todo, el rango de posibles valores de potencia de fuego cae a lo largo de un continuo, no de bloques discretos. Un mejor enfoque es utilizar una fórmula. He aquí un ejemplo:

```
1 setFire(400/distanceTo(e.getX(), e.getY()));
```

Con esta fórmula, a medida que aumenta la distancia del enemigo, la potencia de fuego disminuye. Asimismo, a medida que el enemigo se acerca, la potencia de fuego aumenta. Los valores superiores a 3 se reducen a 3, por lo que nunca dispararemos una bala mayor que 3, pero probablemente deberíamos reducir el valor de todos modos (solo para estar seguros) de esta manera:

```
1 setFire(Math.min(500/distanceTo(e.getX(), e.getY()), 3));
```

### Evitar disparos prematuros

Una situación que encontrarás es que tu Bot dispare antes de haber girado el arma hacia el objetivo. Para evitar disparos prematuros, usa el método `getGunTurnRemaining()` para ver qué tan lejos está tu cañón del objetivo y no dispare hasta que esté cerca.

Además, no puedes disparar si el arma está "caliente" desde el último disparo y llamar a `fire()` o `setFire()` solo desperdiciará un turno. Podemos probar si el arma está fría llamando a `getGunHeat()`.

### Apuntando de manera predictiva

O... "Usar la trigonometría para impresionar a tus amigos y destruir a tus enemigos".

Si quisieramos poder golpear a un robot que recorre las paredes siempre fallaríamos, necesitamos poder predecir dónde estará en el futuro, pero ¿cómo podemos hacerlo?

\$\$ Distancia = Velocidad \* Tiempo \$\$ Usando la anterior formula podemos calcular cuánto tiempo tardará una bala en llegar allí.

- **Distancia:** se puede encontrar llamando a `distanceTo(e.getX(), e.getY())`
- **Velocidad:** según la documentación de RCTR, una bala viaja a una velocidad de: \$\$ 20 - potenciaDeFuego \* 3 \$\$
- **Tiempo:** podemos calcular el tiempo resolviendo: \$\$ Tiempo = \{Distancia \over Velocidad\} \$\$

El siguiente código lo hace:

```

1 // calcular la potencia basado en la distancia
2 double enemyDistance = distanceTo(e.getX(), e.getY());
3 double firePower = Math.min(500 / enemyDistance, 3);
4 // calcular la velocidad de la bala
5 double bulletSpeed = 20 - firePower * 3;
6 // calculamos el tiempo que necesita la bala para impactar al enemigo
7 long time = (long) (enemyDistance / bulletSpeed);
```

### Obteniendo futuras coordenadas X,Y

A continuación, debemos calcular la posición futura de nuestro enemigo:

```

1 // Calcular la velocidad actual de tu enemigo
2 double enemyVelocity = e.getSpeed();
3 // Calcular la dirección actual del enemigo
4 double enemyDirection = e.getDirection();
5 // Descomponer el desplazamiento en las coordenadas X e Y
6 // Componente X del desplazamiento COSENO
7 double deltaX = enemyVelocity * Math.cos(Math.toRadians(enemyDirection));
8 // Componente Y del desplazamiento SENO
9 double deltaY = enemyVelocity * Math.sin(Math.toRadians(enemyDirection));
10
11 // Calcular las coordenadas futuras
12 double futureX = e.getX() + (deltaX * time);
13 double futureY = e.getY() + (deltaY * time);

```

Girando el arma al punto previsto

Ahora debemos apuntar nuestro cañón al lugar previsto de impacto:

```
1 setTurnGunLeft(gunBearingTo(futureX, futureY));
```

Disparando!

Por último disparamos nuestro cañón

```

1 if (getGunTurnRemaining() <= 0 && getGunHeat() == 0) {
2 fire(firePower);
3 }

```

Mejor aún...!

Aquí te dejo algunas mejoras para que las valores:

- ¿Debemos esperar siempre a que el arma esté completamente en la dirección calculada?
- Cuando nuestro enemigo se acerca a una pared, nuestros disparos impactan en ella.
- ¿Puedo saber a quien disparo si mi previsión de impacto falla mucho?

### Investigación y desarrollo propio

A partir de aquí el trabajo será individual de cada alumno (puede solaparse con el paso 3). Deberéis investigar/prever a vuestros adversarios (los conocidos, y los de vuestros compañeros). Aplicar las técnicas que consideréis más útiles para intentar quedar lo más arriba posible en la tabla de clasificación.

⌚14 de julio de 2025

## Entregable de Robocode Tankroyale

---

### Introducción

Robocode es un juego de programación donde el objetivo es codificar un bot en forma de tanque virtual para competir contra otros bots en un campo de batalla virtual. El jugador es el programador del bot, que no tendrá influencia directa en el juego. En cambio, el jugador debe escribir un programa para el cerebro del robot. El programa dice cómo debe comportarse y reaccionar el robot ante los eventos que ocurren en el campo de batalla.

El nombre Robocode proviene de una versión anterior del juego y es una abreviatura de "Código de robot". Con esta nueva versión, se utiliza el mundo "bot" en lugar de "robot".

El juego está diseñado para ayudarte a aprender a programar y mejorar tus habilidades de programación y divertirte mientras lo haces. Robocode también es útil a la hora de estudiar o mejorar el aprendizaje automático (En nuestro caso solo Inteligencia Artificial) en un juego rápido en tiempo real.

Las batallas de Robocode tienen lugar en un campo de batalla, donde pequeños robots tanque automatizados luchan hasta que solo queda uno, como en un juego Battle Royale. De ahí el nombre Tank Royale.

Robocode no contiene sangre, viscera, personas ni política. Las batallas son simplemente por la emoción de la competición que tanto nos gusta.

Documentación del juego: <https://robocode-dev.github.io/tank-royale/>

Repositorio en GitHub: <https://github.com/robocode-dev/tank-royale>

Documentación de la API:

- **Java (JVM)**
- [API overview](#)
- [Bot API](#)
- **.Net**
- [API overview](#)
- [Bot API](#)

Ojo! RCTR se basa en una versión más antigua de RoboCode, con un API diferente, y por lo tanto los robots anteriores, y el funcionamiento del campo de batalla han cambiado sustancialmente.

Por lo tanto, mucha de la documentación que encontrareis es sobre el sistema antiguo, en la que la parte de estrategias es válida, pero no el código que la acompaña. La tarea de traducción del antiguo sistema al nuevo se ha llevado a cabo en forma de un "bridge" entre las dos versiones, y está disponible en GitHub: <https://github.com/robocode-dev/robocode-api-bridge>

En cuanto a la documentación y estrategias, la API antigua todavía se puede encontrar en: <https://robocode.sourceforge.io/docs/robocode/>

Y una página que contaba con muchísima información sobre estrategias, robots, código, etc ya solo está disponible en archive.org, la última versión cacheada que he encontrado disponible está en: <https://web.archive.org/web/20200323061702/http://robowiki.net/>

## Objetivo de la práctica

Usando RoboCode Tank Royale (RCTR) intentaremos ponernos en el papel de los primeros programadores que dotaban de "inteligencia" a los primeros sistemas. Tal y como hemos visto en el apartado de teoría estos sistemas solo reaccionan ante estímulos que previamente haya previsto su programador, carecen de intuición (a no ser que sea simulada) y el resultado de su inteligencia es tan bueno como lo sea su programación.

El profesor establecerá unos robots (simples) que deberemos derrotar para superar la práctica, dotando de cierta "inteligencia" a nuestro robot. No sirve que sea por suerte o de manera aleatoria, debe estar razonada y documentada.

El siguiente paso, una vez aprobado, será una competición entre todo el alumnado para valorar quien ha sido el que mejor Bot ha diseñado, obteniendo así mayor nota que el resto.

## Pasos a seguir

### 1. Preparación del entorno

En una sesión conjunta prepararemos nuestro entorno de trabajo, instalaremos todo lo necesario y haremos algún combate de pruebas.

### 1. Mi primer Bot

Siguiendo la guía de la documentación, y con la ayuda del profesor todo el alumnado generará su primer Bot de muestra, aprenderemos a añadirlo a la batalla y a depurar su funcionamiento.

### 1. ¿Cómo mejoro mi Bot?

Con la ayuda del profesor estudiaremos diferentes estrategias y mejoras que podemos aplicar a nuestro robot para así dotarle de inteligencia.

### 1. Investigación y desarrollo propio

A partir de aquí el trabajo será individual de cada alumno (puede solaparse con el paso 3). Deberéis investigar/prever a vuestros adversarios (los conocidos, y los de vuestros compañeros). Aplicar las técnicas que consideréis más útiles para intentar quedar lo más arriba posible en la tabla de clasificación.

## ¿Qué debo entregar?

A través de la plataforma de AULES todo el alumnado deberá entregar **un archivo ZIP** que contenga:

El código fuente de su Bot (el nombre del bot será el nombre de su autor más los 4 últimos dígitos de su DNI o NIE (sin letras)) y la memoria justificativa en formato PDF.

**El código fuente** del Bot incluye (ejemplo con mi nombre):

- Archivo .java con la clase del Bot (`David4849.java`)
- Archivo .json que incluirá la información completa del autor (`David4849.json`)
- Archivos para inicializar el Bot en Windows y Linux (`David4849.cmd` y `David4849.sh`)

**La memoria** en formato PDF debe contener al menos los siguientes apartados:

- Datos del alumno
- Descripción del funcionamiento: estructura, sistema basado en casos, análisis y evolución de la solución, etc.
- Descripción detallada de los métodos definidos y/o usados
- Conclusiones
- Webgrafía/Bibliografía

## Requisitos mínimos

- **Versión 0.27.0 de la API** (Añadido el 12/12/24, para contemplar la instalación de Docker en Windows.)
- Modo melé
- 10 asaltos
- RamFire, Walls, SpinBot
- Ganar por puntuación acumulada

- Tamaño del campo: 2000 x 2000
- Velocidad de enfriamiento: 0.1
- Máximo tiempo de inactividad: 450
- No llamar a métodos prohibidos
- Demostrar IA (no por azar)

⌚14 de julio de 2025

## 3. UD02

---

### 3.1 Inteligencia Artificial Simbólica

#### Sistemas Basados en Reglas

##### Inteligencia Artificial Simbólica

- **IA simbólica o IA basada en conocimiento:**

- Extraemos conocimiento de expertos y lo representamos de una forma que las máquinas puedan entender.
- Utilizamos este conocimiento para:
  - Resolver problemas automáticamente.
  - Explicar el razonamiento de la máquina.
  - Aprender nuevas cosas.
  - Mejorar el conocimiento existente.

#### REPRESENTACIÓN DEL CONOCIMIENTO

- Conocimiento \vs\ datos \vs\ información:

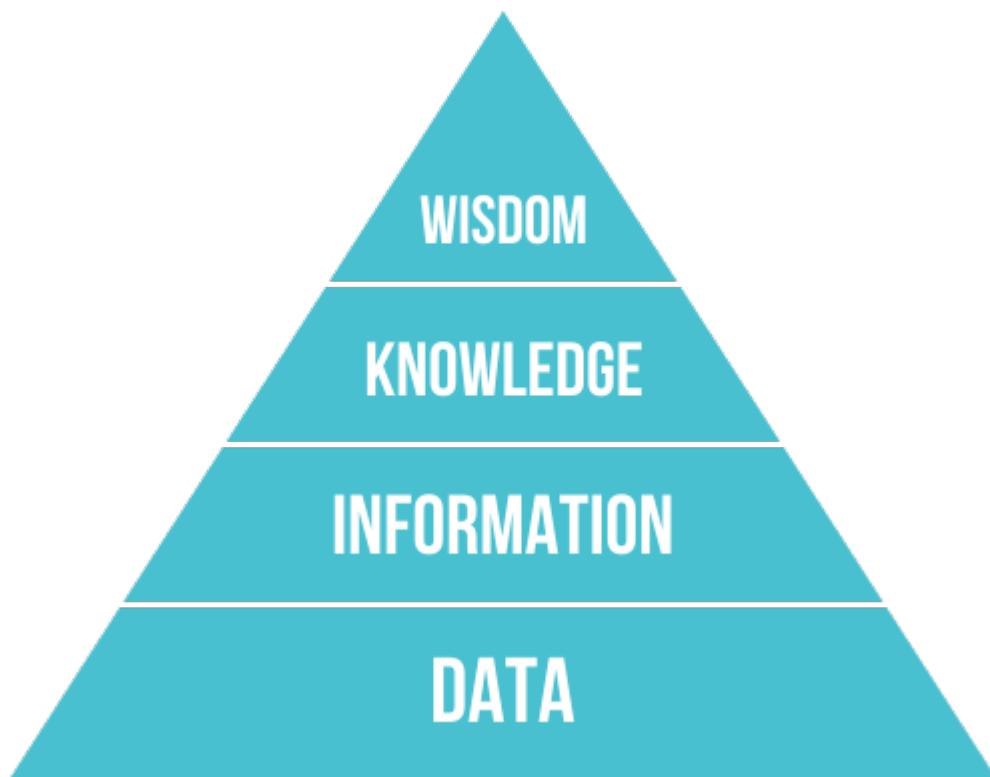
- **Datos:** Hechos o valores.

- **Información:** Datos con significado.

- **Conocimiento:** Información con significado y estructura.

El conocimiento es un conjunto de información estructurada e interrelacionada que permite a un agente realizar tareas.

#### JERARQUÍA DEL CONOCIMIENTO

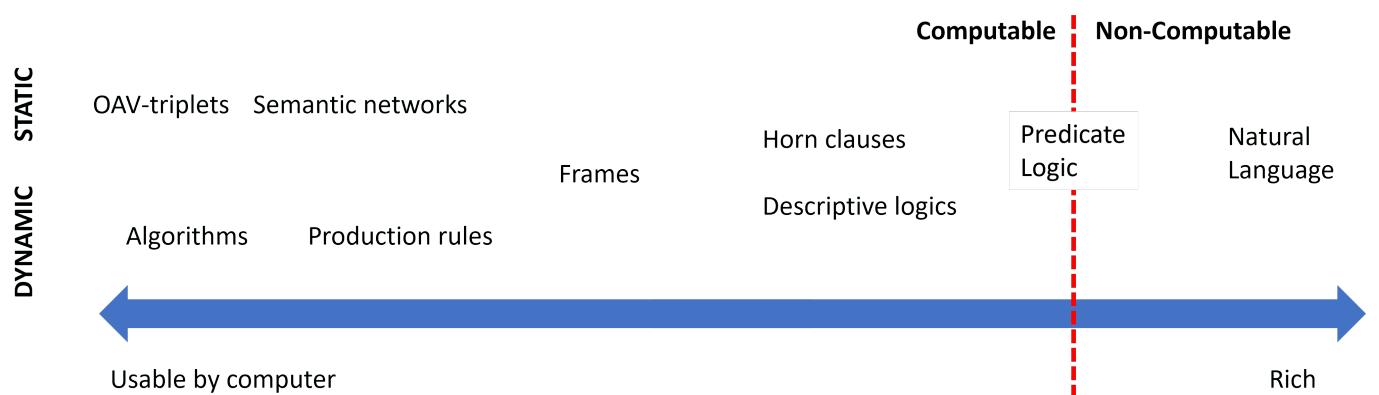


- Muchas veces definimos el conocimiento en relación a conceptos similares.
- La jerarquía del conocimiento o jerarquía de **DIKW** es un modelo que muestra la relación entre *datos, información, conocimiento y sabiduría*.
- **Datos** (\*\*D\*\*ata): Hechos o valores registrados en un soporte físico. Es independiente del agente y puede ser interpretado de distintas formas.

- Ejemplo: "Un reloj inteligente registra la temperatura corporal de la persona."
- **Información** (\*\*I\*\*nformation): Es como los datos son interpretados por un agente. Es subjetiva y depende del agente.
- Ejemplo: "La temperatura corporal de la persona es 37°C"
- **Conocimiento** (\*\*K\*\*nowledge): Es información integrada en nuestro modelo del mundo. Depende del agente y de sus conocimientos previos.
- Ejemplo: "Si la temperatura es superior a 37°C, entonces la persona tiene fiebre"
- **Sabiduría** (\*\*W\*\*isdom): Representa el meta-conocimiento: conocimiento sobre cómo y cuándo aplicar el conocimiento.
- Ejemplo: "Si la persona tiene fiebre, entonces debe tomar paracetamol"

## REPRESENTACIÓN DEL CONOCIMIENTO

- Es la forma en la que representamos el conocimiento para que las máquinas puedan entenderlo.
- Es uno de los problemas fundamentales de la inteligencia artificial.
- Se debe representar de forma que:
- Sea **entendible** para las máquinas.
- Sea **útil** para resolver problemas.
- Sea **eficiente** para ser procesado por las máquinas.
- Podemos ver las diferentes representaciones como un **continuum**:
- A la izquierda tenemos las representaciones más **simples** (algoritmos); utilizables por los ordenadores de forma eficiente pero muy poco flexibles.
- A la derecha tenemos las representaciones más **flexibles** (texto natural); muy potentes pero no utilizables directamente por las máquinas.

**Representaciones de red:**

- En la mente humana el conocimiento se representa como una red de conceptos interrelacionados.
- Las representaciones de red intentamos hacer lo mismo en un grafo dentro de los ordenadores.
- Las llamamos **redes semánticas**.
- Hay diferentes tipos: Pares de atributos y valores, representaciones jerárquicas, representaciones procedurales, lógica, etc.

## PARES DE ATRIBUTOS Y VALORES O TRIPLETES OBJETO-ATRIBUTO-VALOR

- Aprovechamos que un grafo se puede representar como una lista de nodos y aristas para representar el conocimiento.
- El conocimiento se representa como una lista de pares de atributos y valores.
- "El perro es un animal, el perro tiene cuatro patas, el perro tiene pelo, el perro tiene cola, etc."
- "La paloma es un animal, la paloma es un pájaro, la paloma tiene dos patas, etc."
- "El coche es un vehículo, el coche tiene cuatro ruedas, el coche tiene un motor, etc."

## REPRESENTACIONES JERÁRQUICAS

- El conocimiento se representa como un árbol.
- Los nodos del árbol representan conceptos.
- Las aristas representan relaciones entre conceptos.
- Animales → Vertebrados → Mamíferos → Perros → Caniche
- Animales → Vertebrados → Pájaros → Palomas → Paloma común

- Objetos  $\rightarrow$  Vehículos  $\rightarrow$  Coches  $\rightarrow$  Coche de gasolina

#### REPRESENTACIONES PROCEDURALES

- El conocimiento se representa como un conjunto de acciones que se pueden realizar cuando se dan ciertas condiciones.
- Llamamos **reglas de producción** a las **declaraciones** que nos permiten obtener conclusiones a partir de ciertas premisas.
- Son de la forma: **IF** (premisa) **THEN** (conclusión)
- **IF** (la temperatura es superior a 37°C) **THEN** (la persona tiene fiebre)
- **IF** (la persona tiene fiebre) **THEN** (la persona debe tomar paracetamol)

#### LÓGICA

- La lógica es un sistema formal que nos permite representar el conocimiento y razonar sobre él.
- La propuso Aristóteles hace más de 2000 años como herramienta para la **deducción**.
- La lógica proposicional es un sistema formal que nos permite representar el conocimiento y razonar sobre él.
- A nivel teórico es muy potente pero no es directamente utilizable por las máquinas.
- Un subconjunto de la lógica es utilizable en sistemas como prolog.
- Ej:  $\neg(p \rightarrow q)$ : "La persona tiene fiebre",  $\neg(q \rightarrow p)$ : "La persona debe tomar paracetamol"
- $(p \rightarrow q) \rightarrow (\neg p \rightarrow \neg q)$ : "Si la persona tiene fiebre, entonces la persona debe tomar paracetamol"
- $(p \wedge q) \rightarrow p$ : "La persona tiene fiebre y la persona debe tomar paracetamol"

#### Sistemas Expertos

Los sistemas expertos son una aplicación de la inteligencia artificial que hacen uso de conocimientos especializados previamente adquiridos por el ser humano. Los sistemas expertos comenzaron su desarrollo en la década de 1970 y fueron muy populares tanto en esa década como en los años 80 del siglo pasado.

Se considera que los primeros sistemas de inteligencia artificial que fueron capaces de obtener resultados con utilidad práctica fueron los expertos. Se trata de sistemas basados fundamentalmente en reglas. Para el desarrollo de un sistema experto, resulta imprescindible disponer del conocimiento de un especialista en el campo objeto de estudio. Es decir, es necesario contar con información relativa a cómo un especialista trataría el problema propuesto. A los sistemas expertos se les denomina también por ese motivo «sistemas basados en conocimientos», o «sistemas basados en reglas».

Todo sistema experto ha de tener la capacidad de explicar cuál es la decisión que ha tomado.

Un sistema experto se puede definir como un software que es capaz de simular el proceso de decisión que tomaría un experto humano en cierto campo. Por tanto, los sistemas expertos se diseñan de manera que puedan tomar de forma automática decisiones como si fueran expertos. Además, cabe señalar que todo sistema experto debe ser capaz de explicar la decisión que ha tomado y también ha de ser capaz de aprender cuando se le facilita nueva información.

#### ESTRUCTURAS ELEMENTALES DE LOS SISTEMAS EXPERTOS

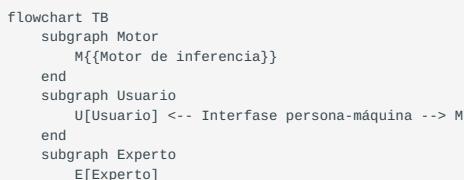
La arquitectura más común de los sistemas expertos es la del sistema basado en reglas. Este tipo de sistemas emplea expresiones del tipo:

«SI ... ENTONCES»

Cada regla representa una porción del conocimiento que se pretende introducir en el sistema. Un conjunto de reglas relacionadas puede llevar de una serie de hechos y datos conocidos hasta algunas conclusiones de utilidad.

Todo sistema experto está formado por los siguientes elementos:

- Interfaz de usuario y de comunicación externa.
- Base de datos de conocimiento.
- Motor de inferencias.
- Sistema para la explicación de las decisiones tomadas.
- Sistema para la adquisición de nuevo conocimiento.



```

E <-- Interfase --> A[Adquisición de conocimiento]
A --> B(Base de conocimientos)
BD(Base de datos)
end
B <--> M
BD <--> M

```

#### Interfaz de usuario y de comunicación externa

Es el medio o vía para las consultas. Debe facilitar una comunicación lo más natural para el usuario, ser sencilla de aprender a utilizar y alertar de posibles datos erróneos de entrada. Los resultados deben ser claros y comprensibles para el usuario. Para conseguir esto, lo habitual ha sido contar con herramientas de desarrollo de interfaces gráficas, e implementar un módulo de comunicaciones y otro de explicaciones.

El módulo de comunicaciones está más enfocado en la interacción con otros sistemas, concretamente, en los casos de automatización de tareas o procesos, como en el caso de robótica industrial.

El módulo de explicación ayuda al ingeniero de conocimiento a refinar el motor de inferencia y al experto a verificar la coherencia de la base de conocimiento. Por otro lado, es el módulo que se encarga de mostrar al usuario el proceso aplicado a la resolución del problema o consulta.

En todo sistema experto resulta importante disponer de una interfaz de usuario que permita una comunicación cómoda del mismo con la aplicación. Toda interfaz hará uso bien de texto, de gráficos o de una combinación de ambos.

Dadas las características fundamentales comunes a todo sistema experto, es imprescindible que el usuario pueda responder de manera cómoda a las preguntas que el sistema le plantee a lo largo del proceso de resolución del problema.

Además, las conclusiones alcanzadas por el sistema serán mostradas al usuario a través de la interfaz.

Dentro de la interfaz se ha de tener en cuenta la parte dedicada a la comunicación externa, dado que resulta altamente probable que el sistema tenga que hacer uso de datos externos al mismo.

#### Base de datos de conocimiento

Contiene el conocimiento y la experiencia de los expertos en un campo determinado, estructurado y codificado, preparado para entregar dicho conocimiento cuando sea requerido por el sistema. Ha sido generado a partir de las referencias dadas por los expertos en dicho campo.

El conocimiento puede estar organizado mediante listas, descripción de objetos relacionados con el problema en estudio, cálculo de predicados, redes semánticas y las relaciones o reglas de producción entre ellos. También se considera importante que estén los procedimientos de aplicación de dicho conocimiento en función del problema a resolver.

Para la creación de la base de datos de un sistema experto se necesitará contar con la participación de personas con experiencia que sean capaces de codificar sus conocimientos como reglas de la forma:

```

1 SI <antecedente> ENTONCES <consecuencia>

```

En algunos casos, una regla puede tener múltiples antecedentes que se unen mediante conectores como O e Y. Este tipo de regla se denomina regla compuesta.

La estructura de una regla compuesta es la que se muestra a continuación:

```

1 SI <antecedente1> Y <antecedente2>
2 ENTONCES <consecuencia>
3 SI <antecedente1> O <antecedente2>
4 ENTONCES <consecuencia>

```

Por tanto, las reglas constituyen la forma más común de codificar el conocimiento adquirido por un experto.

Se pretende construir un sistema experto que decida si se concede o no un crédito al consumo a cierto cliente. Este sistema experto debe trabajar como lo haría un bancario acostumbrado a dicha operación.

Un bancario con experiencia indica que únicamente se conceden este tipo de créditos a mayores de 18 años que dispongan de nómina y cuyo contrato sea bien de carácter indefinido o que la duración del mismo sea superior al tiempo necesario para la devolución de todas las cuotas del préstamo.

Este conocimiento se podría expresar como:

```

SI (cliente mayor de 18 años) SI (cliente mayor de 18 años) Y (tiene nómina) Y (tiene contrato indefinido) O (la duración del
contrato es superior al tiempo de devolución del préstamo) ENTONCES (conceder préstamo solicitado)

```

Por tanto, si el único criterio que se emplea para la concesión de créditos al consumo es el indicado por el bancario, un sistema experto que tenga implementada la regla expuesta más arriba será capaz de gestionar la concesión de créditos obteniendo los mismos resultados.

A la hora de implementar un sistema experto, resulta conveniente tener en cuenta que el conocimiento que proporcionan los expertos se pueda clasificar en distintas categorías:

- En primer lugar, un tipo de conocimiento muy utilizado es el denominado conocimiento procedimental. Este conocimiento se refiere a la realización de alguna tarea que se lleva a cabo con el fin de, por ejemplo, mejorar el rendimiento de un sistema o de un proceso. Así, siempre que se disponga de conocimiento relativo a cómo resolver un problema paso a paso, dicho conocimiento se denominará procedimental.
- Existe también otro conocimiento que es del tipo objetivo y que se encuentra en los libros y manuales de una especialidad. Es el llamado conocimiento factual. Si bien es accesible por otras vías, resulta de utilidad implementarlo en un sistema experto dada la rapidez de acceso al mismo.
- Además, también hay otra clase que es la propia de cada experto, que no está completamente basada en hechos objetivos y que no se encuentra en los libros de texto. Este conocimiento se denomina conocimiento heurístico y su implementación en sistemas expertos resulta muy adecuada. Si se alimenta a un sistema experto con conocimientos de tipo heurístico, se puede conseguir que, por ejemplo, personal destinado a la realización de una tarea y con escasa experiencia en la misma tome decisiones similares a las que elegiría un experto.
- La representación del conocimiento en un sistema experto por reglas del tipo SI ... ENTONCES contribuye a hacer más sencilla su explicación pues son fácilmente entendibles tanto por los programadores del sistema como por sus usuarios. Nótese también que la base de datos de conocimiento contiene la información que empleará el motor de inferencia.

#### Base de hechos o datos

Es la memoria de trabajo propiamente dicha. Consiste en una memoria temporal auxiliar que almacena variables de inicio, valores de variables intermedias y las variables de salida de la consulta.

En esta unidad, queda registrado todo el histórico de estados del sistema en la consulta.

Durante una consulta, el usuario introduce la información que se tiene del problema actual en la base de hechos y el sistema sincroniza ésta con el conocimiento que hay disponible al respecto en la base de conocimiento, de forma que se puedan deducir nuevos hechos. Para esto es necesario que las base de datos sean de tipo relacional.

#### Motor o mecanismo de inferencia

Es la unidad lógica que aplica las reglas sobre la base de conocimientos a partir de las consultas, extrayendo conclusiones. Utiliza un método fijo de solución de problemas configurado imitando el proceso humano de los expertos para resolver ese tipo de problemas.

El motor de inferencias es el elemento del sistema experto encargado de realizar el razonamiento. Así, el motor de inferencias es capaz de generar nueva información a partir del contenido existente en la base de datos y, por tanto, de tomar decisiones y contribuir a la resolución de problemas reales.

El motor de inferencia determina las acciones que tendrán lugar, el orden en el que lo harán y la interacción entre las distintas partes del sistema. También selecciona las reglas a aplicar y determina cómo y cuándo se van a aplicar las reglas programadas. Finalmente, también se encarga de la interacción con el usuario.

#### Sistema para la explicación de las decisiones tomadas

Una vez que el motor de inferencias ha llegado a una decisión, resulta de gran importancia que el sistema sea capaz de explicárselo de forma conveniente al usuario. Una manera de hacerlo es mostrando las reglas de inferencia que el sistema empleó en su proceso de razonamiento. Sin embargo, este método podría ser, en función de la aplicación de que se trate, sumamente tedioso.

Por tanto, todo sistema experto debe disponer de algún tipo de subsistema que permita presentar una explicación de las decisiones tomadas de manera que resulte comprensible para el usuario.

#### Sistema para la adquisición de nuevo conocimiento

Es la parte del sistema que facilita la estructuración, implementación y actualización del conocimiento en las bases de datos. La clave es que sea una herramienta que se pueda utilizar sin tener un perfil especialmente técnico y sin tener que programar, aunque sí que debe permitir el acceso a través de código.

Se entiende por sistema de adquisición de nuevo conocimiento una interfaz que permita que un experto en el campo sea capaz de introducir nueva información en el sistema. Dada la naturaleza de los sistemas expertos, es necesario que, una vez puesto en operación, resulte posible seguir añadiendo nueva información a medida que sea necesario y, para ello, es imprescindible disponer de algún sistema que permita la adquisición de este conocimiento.

**DINÁMICA DE UN SISTEMA EXPERTO.**

El objetivo de los sistemas basados en el conocimiento es hacer que la información crítica requerida para que el sistema funcione sea explícita en lugar de implícita. En un programa informático tradicional, la lógica está incrustada en un código que, por lo general, solo puede ser revisado por un especialista informático. Con un sistema experto, el objetivo era especificar las reglas en un formato que fuera intuitivo y fácil de entender, revisar e incluso editar por expertos en el dominio en lugar de expertos en TI. Los beneficios de esta representación del conocimiento explícita fueron el desarrollo rápido y la facilidad de mantenimiento.

Los sistemas expertos, con su capacidad para combinar información y reglas de actuación, han sido vistos como una de las posibles soluciones al tratamiento y recuperación de información, no sólo documental. La década de 1980 fue prolífica en investigación y publicaciones sobre experimentos de este orden, interés que aún no ha disminuido.

Lo que diferencia a este tipo de sistemas de un sistema tradicional de recuperación de información es que este último sólo es capaz de recuperar lo que existe explícitamente, mientras que un sistema experto debe ser capaz de generar información no explícita, razonando con los elementos que se le dan. Pero la capacidad de los sistemas expertos en el ámbito de la recuperación de la información no se limita a la recuperación. Pueden utilizarse para ayudar al usuario, en selección de recursos de información, en filtrado de respuestas, etc. Un sistema experto puede actuar como un intermediario inteligente que guía y apoya el trabajo del usuario final.

Veamos ahora los tipos de sistemas expertos que se desarrollaron y cómo resolvieron las tareas clave que permitieron su funcionamiento.

## Mecanismos de razonamiento.

Los principales mecanismos o modos de razonamiento son:

- Encadenamiento hacia delante: se parte de hechos para llegar a resultados.
- Encadenamiento hacia atrás: se parte de los resultados y se trata de encontrar o volver a los hechos.
- Encadenamiento mixto: combina los anteriores.
- Algoritmo de búsqueda heurística: el proceso de inferencia es una búsqueda en una estructura de tipo árbol.
- Herencia: usado en entornos de programación orientada a objetos. Un objeto hijo hereda propiedades y hechos de los padres.

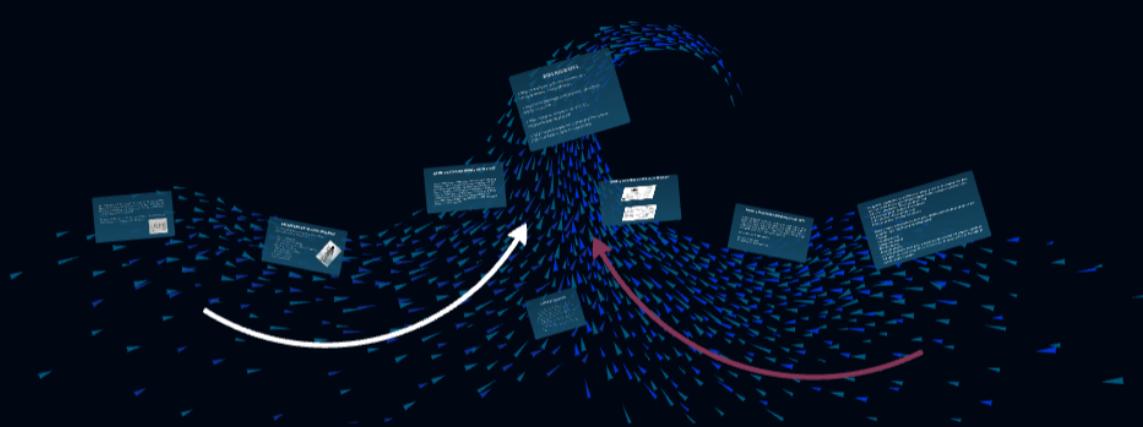
Para obtener conclusiones, utilizaremos los diferentes tipos de reglas y estrategias de inferencia y control. Te recomendamos empezar por considerar las más básicas como son **Modus Ponens** y **Modus Tollens** como sistemas de inferencia y el **encadenamiento de reglas hacia adelante** y **encadenamiento de reglas hacia atrás** como estrategias de inferencia.

Para conocer mejor los sistemas de inferencia **Modus Ponens** y **Modus Tollens**, puedes recurrir a sus correspondientes artículos en la Wikipedia:

- **Modus Ponens** "si  $P$  implica  $Q$ ; y si  $P$  es verdad; entonces  $Q$  también es verdad."
- **Modus Tollens** "Si  $P$  implica  $Q$ , y  $Q$  no es cierto, entonces  $P$  no es cierto"

Así como ver este [vídeo](#) corto en el que se hace un planteamiento sencillo de los conceptos.

Encadenamiento hacia adelante y hacia atrás



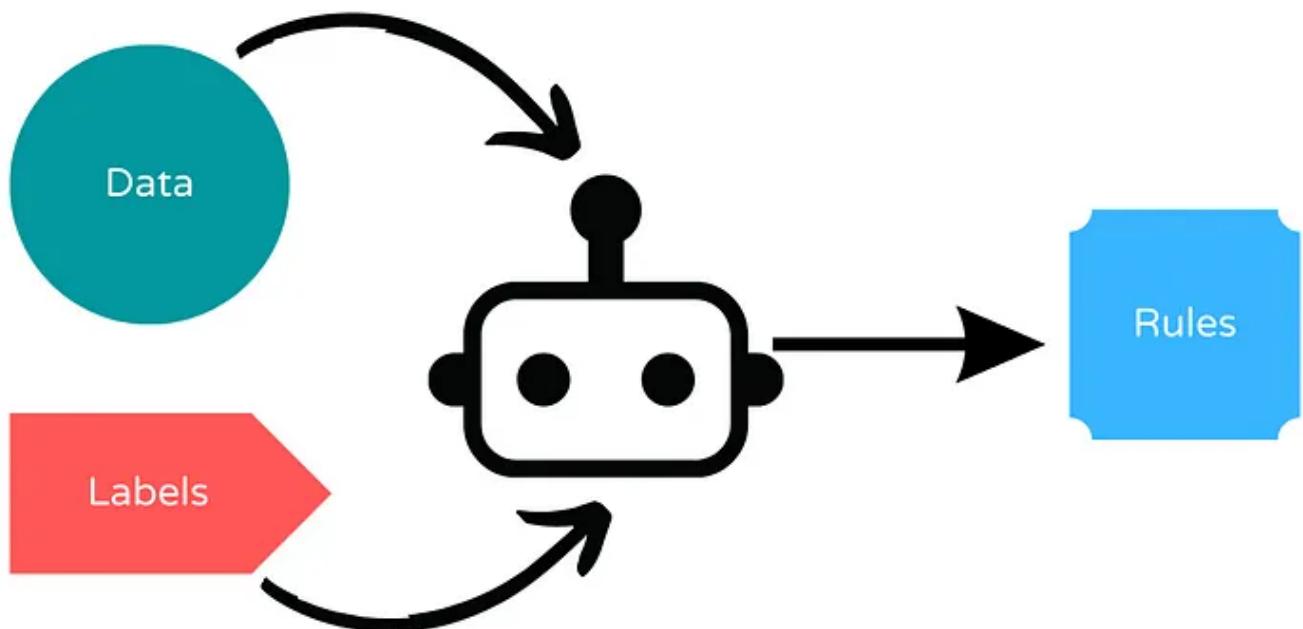
**ENCADENAMIENTO HACIA ADELANTE Y HACIA ATRAS**

Adriana Rodriguez  
Marcela Herrera

 Prezi

#### Sistemas híbridos Reglas/Datos

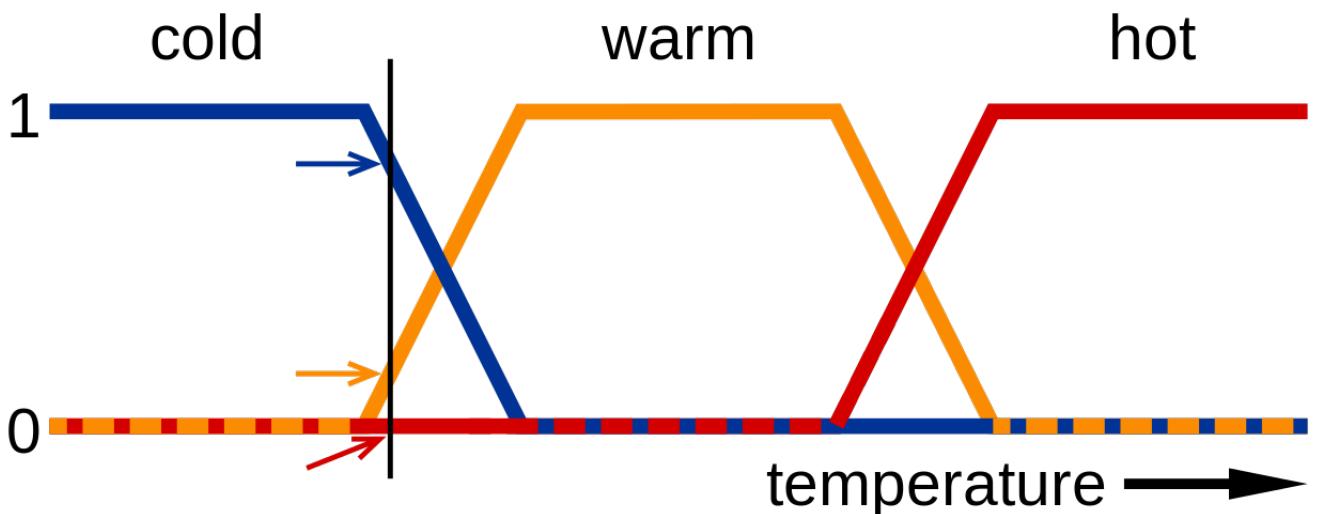
- Dos enfoques:
- Deducción de reglas a partir de datos.
- Facilita la **interpretación** del razonamiento.
- Integración de reglas definidas por el usuario y Aprendizaje Automático.
- Permite definir unas reglas que se pueden **mejorar** con el aprendizaje automático.



## LIBRERÍAS

- [Human-Learn](#):
- Permite definir y dibujar reglas que se pueden mejorar con el aprendizaje automático.
- [skope-rules](#):
- Analiza los datos y deduce reglas para clasificar.
- Permite analizar las reglas para mejorarlas e interpretarlas.
- [SpaCy](#):
- Permite definir reglas para la extracción de información para textos.
- Útil en casos donde no se dispone de suficientes datos etiquetados o por casos específicos.

## Sistemas de razonamiento impreciso

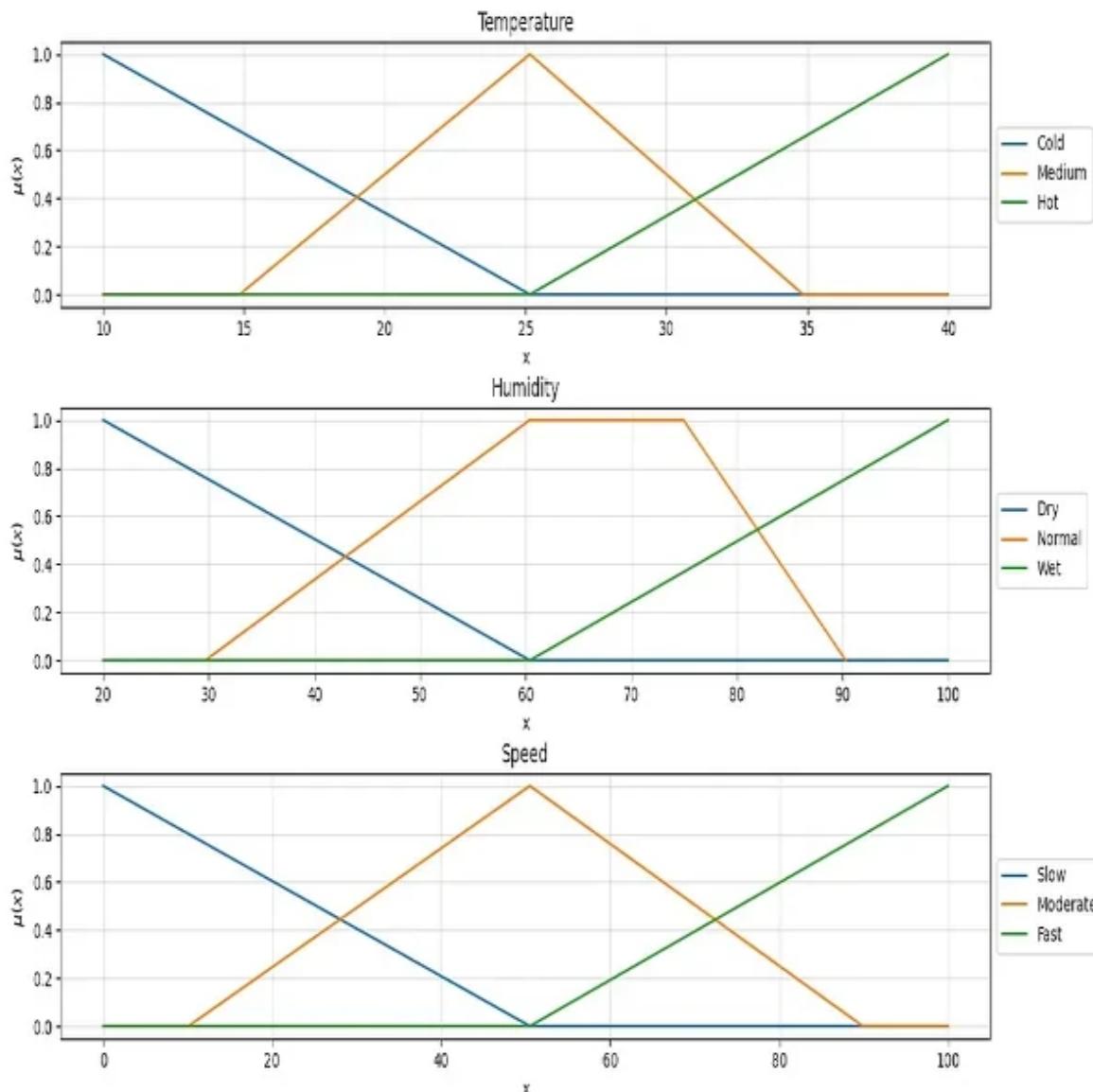


## DEFINICIÓN

- **Lògica difusa o lògica borrosa:**
  - Extensión de la lògica proposicional para trabajar con la incertidumbre.
  - Permite trabajar con valores imprecisos.
- **Sistemas de razonamiento impreciso:**
  - Sistemas basados en reglas que utilizan la lògica difusa.
  - Permiten trabajar con valores **continuos**.
  - Facilitan modelar el **conocimiento humano**.
  - Muy apropiados para **sistemas de control**
  - Nos permiten tener una **buenasolución**, si no la **mejor**.

## LÓGICA DIFUSA

- La lògica proposicional es **binaria**.
- Un enunciado es **cierto** o **falso**.
- La lògica difusa permite trabajar con valores **continuos**.
- Un enunciado puede ser **cierto i falso** en un grado **parcial**.
- Los valores de verdad son **números reales** en el intervalo  $([0, 1])$ .
- $\lambda(0: \text{Falso}), \lambda(1: \text{Cierto}), \lambda(0.5: \lambda(\text{Cert}))$  en un  $(50\%)$
- La pertenencia de un elemento a un conjunto vendrá dada por una **función de pertenencia**.
- $\lambda(\mu_A(x))$ : Grado de pertenencia de  $\lambda(x)$  al conjunto  $\lambda(A)$ .



- La lógica difusa facilita la **representación del conocimiento humano**.
- Los humanos no razonamos en términos binarios.
- Los humanos no tenemos un conocimiento preciso ni completo.
- Conceptos como  $\backslash(\text{húmedo})$  o  $\backslash(\text{frio})$  son difíciles de definir con precisión.
- La lógica difusa nos permite definirlos con **funciones de relevancia**.
- El poder trabajar con estos conceptos facilita la creación de dispositivos como secadores o termostatos.
- *"Si la temperatura es fría, entonces enciende la calefacción"*

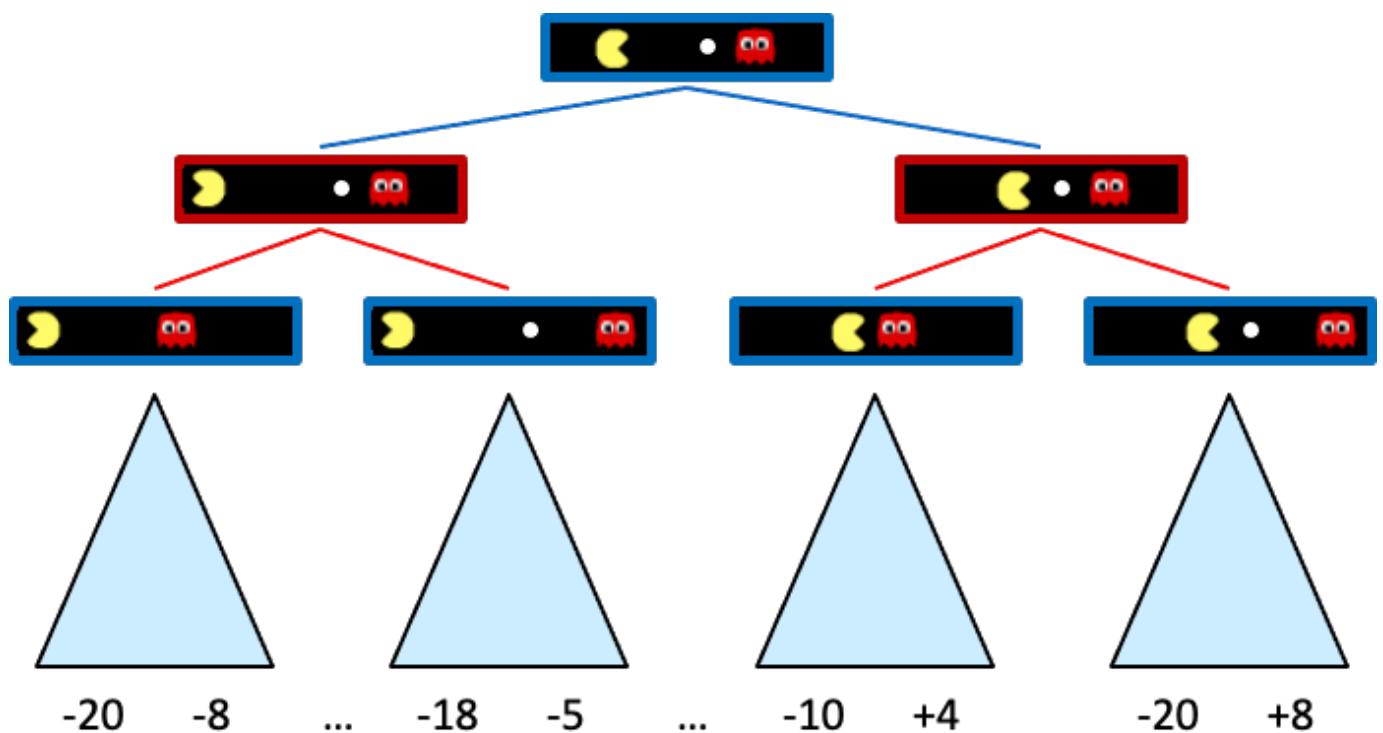
#### CONCEPTOS BÁSICOS

- **Variable lingüística:** Variable que puede tomar valores lingüísticos.
- Ejemplos:  $\backslash(\text{Temperatura})$
- **Valores lingüísticos:** Valores que puede tomar una variable lingüística.
- Ejemplo:  $\backslash(\text{Frío}, \text{Calor})$
- **Función de pertenencia:** Función que asigna a cada valor de una variable lingüística un grado de pertenencia a un valor lingüístico.
- Ejemplo:  $\backslash(\text{Temperatura} = 27^{\circ}\text{C} \rightarrow \text{Calor} = 0.8, \backslash(\text{Combustible}, \text{Calor} = 0.2))$
- **Regla difusa:** Regla que utiliza valores difusos.

- Ejemplo: "Si la temperatura es **fría**, entonces **calefacción alta**"
- **Función de agregación:** Función que combina los valores difusos de las reglas para deducir la conclusión final.
- Exemple:  $(\text{Calor} = 0.8, \text{Humedo} = 0.7 \rightarrow \text{Sensación}) : \text{desgradable} = 0.8$ )
- **Sistema de razonamiento impreciso:**
- Sistema basado en reglas que utiliza la lógica difusa.
- Ejemplo: Sistema de control de temperatura de una casa.

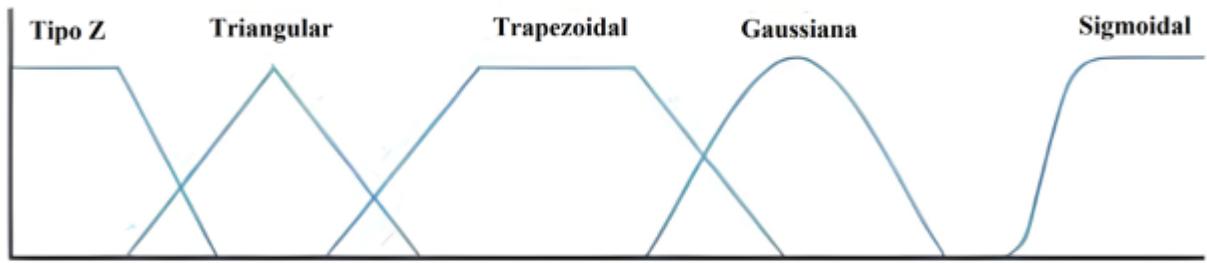
## FUNCIONAMIENTO DE LOS SISTEMAS DE RAZONAMIENTO IMPRECISO

- *Fuzzyfication:*
- Conversión de los datos de entrada precisos a valores difusos.
- Pasamos de valores precisos a valores difusos.
- Utiliza las **funciones de pertenencia**.
- Asigne a cada valor de entrada un grado de relevancia para cada **variable de idioma**
- $(27^\circ\text{C} \rightarrow \text{Calor} = 0.8, \text{Mucha} : \text{calor} = 0.2)$
- *Evaluación de las reglas:*
- En este paso se **aplican las reglas del sistema**.
- Se establece la relación entre las **variables de entrada** y las **variables de salida**.
- "Si la temperatura es **alta** y la humedad es **baja**, entonces la velocidad del ventilador debe ser **alta**"
- Las **funciones de relevancia** de las **variables de entrada** se combinan
- para deducir la **relevancia** de la variable de **salida**.



- *Desfuzzyfication:*
- Conversión de los datos de salida difusos a valores precisos.
- Pasamos de valores difusos a valores precisos.
- Utiliza las **funciones de agregación**.
- Combina las conclusiones de las reglas para deducir la conclusión final.
- Se suele utilizar la función de **centro de gravedad o máximo**.

## FUNCIONES DE PERTENENCIA

**Funciones de pertenencia**

- Las más utilizadas son las **funciones trapezoidales** y las **funciones triangulares**.
- Las sinusoidales son útiles para representar **periodos**.
- Las sigmoidales son útiles para representar **probabilidades**.

## EJEMPLO: PROPINAS

## Variables d'entrada

Utilizaremos funciones triangulares para representar las variables de entrada y salida

- **Servicio:**
- **Bajo:**  $\lambda([0, 5])$
- **Media:**  $\lambda([0, 10])$
- **Alta:**  $\lambda([5, 10])$

