



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

Indice de contenidos

1. UD00	5
1.1 Información importante	5
Denominación del curso	5
Contenidos	5
Resultados de Aprendizaje (RA)	5
Legislación vigente	6
Evaluación	6
2. Docker	7
2.1 Introducción a docker	7
Generalidades	7
Instalación	10
Uso	13
Opciones	20
Casos de uso	22
Copias de seguridad	27
Contenedores ejemplo	28
Ejercicios	29
Tarea entregable	29
3. UD01	30
3.1 Caracterización de sistemas y utilización de modelos de Inteligencia Artificial	30
Fundamentos de los Sistemas Inteligentes	30
Tipos de Inteligencia Artificial. Escuelas y clasificaciones	43
Utilización de modelos de Inteligencia Artificial	48
Técnicas de la Inteligencia Artificial	50
Campos de Aplicaciones de la Inteligencia Artificial	55
Nuevas Formas de Interacción	68
Mapa conceptual	70
3.2 Diapositivas de la UD01	72
3.3 Actividades	73
Robocode TankRoyale	73
Entregable de Robocode Tankroyale	84
3.4 Talleres	87
Taller UD01_T01: Preparar entorno para Java	87
Taller UD01_T02: Crear cuenta en GitHub	91
Taller UD01_T03: Markdown	98

4. UD02	107
4.1 Sistemas basados en el conocimiento	107
Inteligencia Artificial Simbólica	107
Sistemas Expertos	109
Sistemas híbridos Reglas/Datos	113
Sistemas de razonamiento impreciso	114
4.2 Diapositivas de la UD02	123
4.3 Actividades guiadas de la UD02	124
4.4 Actividades entregables de la UD02	125
5. UD03	126
5.1 Procesamiento del Lenguaje Natural	126
Introducción al PLN	126
Técnicas de procesamiento de lenguaje. Limitaciones	129
Formación del investigador en PLN	134
Elaboración de un sistema de procesamiento de lenguaje orientado a una tarea específica	135
5.2 Diapositivas de la UD03	137
5.3 Actividades guiadas de la UD03	138
5.4 Actividades entregables de la UD03	139
6. UD04	140
6.1 Análisis de sistemas robotizados	140
Robots	140
Hardware de Robots	140
¿Qué tipo de problema resuelve la robótica?	143
Percepción robótica	144
Planificación y Control	146
Planificación de movimientos inciertos	151
Aprendizaje por refuerzo en robótica	151
Humanos y robots	151
Dominios de aplicación	154
Resumen	155
6.2 Actividades guiadas de la UD04	156
6.3 Actividades entregables de la UD04	157
7. UD05	158
7.1 Aplicación de principios legales y éticos de la IA	158
Introducción	158
Deontología profesional en inteligencia artificial	158
La privacidad de los datos	159
Protección frente a errores	160

Principios éticos	161
Sesgos en el desarrollo y aplicación de la IA y el Big Data	174
Regulación Europea sobre IA	176
7.2 Actividades guiadas de la UD05	177
7.3 Actividades entregables de la UD05	178
Relación de Legislación y Ética en IA con Series, Películas y Libros	178
8.  Sobre mí...	181
8.1  David Martínez Peña	181
8.2  Contacto:	181
9. Fuentes de información	182

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
	📅 17 PRIMER TRIMESTRE	45
1AVA	❖ UD01: Caracterización de sistemas y utilización de modelos de Inteligencia Artificial ❖ UD02: Sistemas Expertos ❖ UD03: Procesamiento del Lenguaje Natural	15 15 15
	📊 1a EVALUACIÓN	2
	📅 17 SEGUNDO TRIMESTRE	45
2AVA	🤖 UD04: Análisis de sistemas robotizados ⚖ UD05: Aplicación de principios legales y éticos de la Inteligencia Artificial Proyecto intermodular	15 10 20
	📊 2ª EVALUACIÓN	2
🎓 CONVOCATÒRIA ORDINÀRIA		2
	📊 TOTAL	90

🎯 Resultados de Aprendizaje (RA)

Descripción	Unidades	Nota mínima	Peso
RA1	1	5	10,00%

	Descripción	Unidades	Nota mínima	Peso
RA2	Utiliza modelos de sistemas de Inteligencia Artificial implementando sistemas de resolución de problemas.	1	5	10,00%
RA3	Relaciona el procesamiento de lenguaje natural con sus aplicaciones determinando su potencial e identificando sus limitaciones.	3	5	15,00%
RA4	Analiza sistemas robotizados, evaluando opciones de diseño e implementación.	4	5	15,00%
RA5	Aplica sistemas expertos evaluando la influencia de los controladores inteligentes en el comportamiento del sistema.	2	5	15,00%
RA6	Aplica principios legales y éticos al desarrollo de la Inteligencia Artificial integrándolos como parte del proceso.	5	5	15,00%
RA7	Proyecto intermodular		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.

⌚ 13 de noviembre de 2025

2. Docker

2.1 Introducción a docker

Generalidades

Imprescindibles

- [¿Qué es docker? Imagen vs contenedor](#)



- [Curso de docker](#)



- Creación de imágenes



Arquitectura



Imágenes

- ▶ Plantilla de solo lectura
- ▶ Sistema de ficheros y parámetros listos para ejecutar
- ▶ Basadas en sistemas operativos Linux

Contenedores

- ▶ Instancia de una imagen
- ▶ Es un directorio dentro del sistema, similar a los “jail root”
- ▶ Pueden ser ejecutados, reiniciados, parados...

Instalación

Instalación en Ubuntu

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es>

Requisitos previos:

- **apt-transport-https:** permite que el administrador de paquetes transfiera datos a través de https
- **ca-certificates:** permite que el navegador web y el sistema verifiquen los certificados de seguridad
- **curl:** transfiere datos (similar a wget)
- **software-properties-common:** agrega scripts para administrar el software

```
1 sudo apt-get install curl apt-transport-https ca-certificates software-properties-common
```

Agregamos repositorio

```
1 # Primero clave GPG
2 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
3
4 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
5
6 sudo apt update
7
8 sudo apt install docker-ce
9
10 sudo systemctl status docker
```

Por defecto, el comando docker solo puede ser ejecutado por el usuario root o un usuario del grupo docker, que se crea automáticamente durante el proceso de instalación de Docker.

Para evitar escribir sudo al ejecutar el comando docker, agregue su nombre de usuario al grupo docker:

```
1 sudo usermod -aG docker ${USER}
2
3 # Cerramos y abrimos sesión de nuevo o ejecutamos
4 su - ${USER}
5
6 # Confirmamos los grupos de nuestro usuario
7 id -nG
```

Docker Desktop

¿QUÉ ES DOCKER DESKTOP?



Es la aplicación oficial de Docker que te da una **interfaz gráfica (GUI)** para manejar contenedores, además de la línea de comandos.

¿PARA QUÉ SIRVE?

- **Gestión visual:** Ver contenedores, imágenes y volúmenes de forma gráfica
- **Configuración fácil:** Ajustar recursos (CPU, RAM) con sliders
- **Monitorización:** Ver en tiempo real qué está pasando



COMPATIBILIDAD POR SISTEMA OPERATIVO

Windows

- **Windows 10/11** 64-bit (versiones Home, Pro, Enterprise, Education)
- **Requisitos importantes:**
 - Habilitar **WSL 2** (Windows Subsystem for Linux)
 - Virtualización activada en BIOS/UEFI
- **Windows Home** necesita WSL 2, **Pro/Enterprise** puede usar Hyper-V

macOS

- **macOS 12 Monterey** o superior
- **Tipos de chip:**
 - **Apple Silicon** (M1, M2, M3, etc.)
 - **Intel** con procesador de 2010 o más nuevo
- Necesita **macOS actualizado**

Linux (versión nativa)

- **Distribuciones compatibles:**
 - Ubuntu 20.04 LTS o superior
 - Debian 11 o superior
 - Fedora 36 o superior
 - Arch Linux (y derivados)
- **Requisitos:** kernel 5.10+, systemd, 64-bit

GUÍA RÁPIDA DE INSTALACIÓN

Windows:

1. Descarga desde docker.com/products/docker-desktop
2. Ejecuta el instalador `.exe`
3. Sigue el asistente (marca "Use WSL 2" si tienes Windows Home)
4. Reinicia cuando termine

5. ¡Listo! Docker se inicia automáticamente

macOS:

1. Descarga desde la web oficial
2. Arrastra Docker.app a la carpeta Applications
3. Ejecuta desde Launchpad
4. Autoriza con contraseña del sistema
5. Espera a que configure todo (puede tardar unos minutos)

Linux (Ubuntu/Debian ejemplo):

```

1 # Paso 1: Descargar .deb oficial
2 wget https://desktop.docker.com/linux/main/amd64/docker-desktop-4.25.0-amd64.deb
3
4 # Paso 2: Instalar
5 sudo apt install ./docker-desktop-*.deb
6
7 # Paso 3: Iniciar
8 systemctl --user start docker-desktop

```

Problemas con Ubuntu 24.04 ▾

Si tienes problemas con Ubuntu 24.04, yo me he encontrado dos:

1. Problema de permisos

```

1 ...
2 S'estan processant els activadors per a desktop-file-utils (0.27-2build1)...
3 N: La baixada es duu a terme fora de l'entorn segur com a root ja que el fitxer «/home/ubuntu/docker-desktop-4.27.2-amd64.deb» no és accessible per l'usuari «_apt». - pkgAcquire::Run (13: Permission denied)

```

Lo he resuelto cambiando los permisos del deb:

```

1 # Change ownership of the file to make it accessible
2 sudo chown _apt:root /home/ubuntu/docker-desktop-4.27.2-amd64.deb
3 # Or alternatively, change permissions to make it readable
4 sudo chmod 644 /home/ubuntu/docker-desktop-4.27.2-amd64.deb

```

2. Lanzas Docker-desktop, aparece el ícono, pero desaparece y la aplicación no incia: Parece un problema con un cambio en Ubuntu 24.04 que he resuelto con la información de este [post](#):

```

1 sudo sysctl -w kernel.apparmor_restrict_unprivileged_userns=0
2 systemctl --user restart docker-desktop

```

En algunos casos parece que la solución anterior solo sirve hasta que reinicias, si es así, prueba esto también:

Crea un nuevo fichero:

```
1 sudo nano /etc/apparmor.d/opt.docker-desktop.bin.com.docker.backend
```

Escribe dentro el siguiente contenido:

```

1 abi <abi/4.0>,
2
3 include <tunables/global>
4
5 /opt/docker-desktop/bin/com.docker.backend flags=(default_allow) {
6 userns,
7
8 # Site-specific additions and overrides. See local/README for details.
9 include if exists <local/opt.docker-desktop.bin.com.docker.backend>
10 }

```

Reinicia el servicio apparmor.service :

```
1 sudo systemctl restart apparmor.service
```

VENTAJAS DOCKER DESKTOP (GUI) VS LÍNEA DE COMANDOS (CLI)

Ventajas de Docker Desktop:

- **Más fácil para empezar** - Ideal para principiantes
- **Todo integrado** - No necesitas instalar nada más
- **Debugging visual** - Ves los logs y estados de un vistazo
- **Gestión de recursos** - Controlas CPU/RAM fácilmente

Desventajas:

- **Más pesado** - Consume más recursos de tu PC
- **Menos flexible** - Algunas opciones avanzadas solo por comandos
- **Dependes de la GUI** - Si se cierra la app, pierdes la interfaz

Conclusión:

- **Empezad con Docker Desktop** para aprender sin frustraciones
- **Aprended también los comandos básicos** para ser más versátiles
- Usad **ambos**: la GUI para lo cotidiano y la terminal para lo avanzado

Uso

Comandos básicos

GESTIÓN DE IMAGENES

```

1 docker image
2 docker history
3 docker inspect
4 docker save/load
5 docker rmi

```

GESTIÓN DE CONTENEDORES

```

1 docker attach
2 docker exec
3 docker inspect
4 docker kill
5 docker logs
6 docker pause/unpause
7 docker port
8 docker ps
9 docker rename
10 docker start/stop/restart
11 docker rm
12 docker run
13 docker stats
14 docker top
15 docker update

```

EJEMPLO

```

1 # Ver los contenedores que tenemos
2 docker ps
3 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4
5 # Ver las imágenes que tenemos
6 docker images
7 REPOSITORY TAG IMAGE ID CREATED SIZE
8
9 # Crear un contenedor con una imagen básica de debian
10 # Como no tenemos ninguna imagen de debian, la descarga y la ejecuta
11 docker run debian
12 # Intentamos ver el contenedor en ejecución, no aparece nada porque ya se ha cerrado
13 docker ps
14 # Podemos verlo con
15 docker ps -a
16 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
17 09b14daab800 debian "bash" 2 seconds ago Exited (0) 1 second ago
18
19 # Ejecutar un comando en un contenedor
20 docker run debian /bin/echo "Hello World"
21 Hello World
22
23 # Información
24 docker inspect debian

```

Crear contenedor interactivo y con nombre

<https://jolthgs.wordpress.com/2019/09/25/create-a-debian-container-in-docker-for-development/>

Para que docker no se invente un nombre como “pensive_wozniak” (comando anterior) podemos definir el nombre que queremos.

Utilizaremos una de las imágenes de: https://hub.docker.com/_/debian/tags

```

1  # Obtenemos la imagen, en el apartado anterior la hemos ejecutado directamente con "run", esto
2  # la obtiene implícitamente. En este caso la vamos a descargar.
3  $ docker pull debian:13-slim
4
5  # --name
6  # -h hostname que tendrá el contenedor
7  # -e codificación de caracteres
8  # -it modo interactivo
9  # /bin/bash -l la shell que se ejecutará
10 $ docker run --name debian-mini -h equipo1 -e LANG=C.UTF-8 -it debian:13-slim /bin/bash -l
11
12 --- Estamos dentro del contenedor ---
13 # Una vez dentro del contenedor podemos actualizarlo e instalar los paquetes que creamos necesarios
14 apt update && apt upgrade --yes && apt install sudo locales --yes
15 # Configurar timezone
16 dpkg-reconfigure tzdata
17
18 # Vamos a nuestra home y creamos un archivo
19 cd
20 echo "holo" > prueba.txt
21
22 # Salimos del contenedor
23 exit (o control + d)
24
25 --- Ahora volvemos a la consola de nuestro equipo ---
26 $ docker ps
27 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
28
29 $ docker ps -a
30 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
31 8c6b29c818ee  debian:13-slim "/bin/bash -l" 44 seconds ago Exited (0) 11 seconds ago
      debian-mini

```

Imágenes

Es un instalador donde podemos incorporar nuestra aplicación. Es el punto de inicio para crear contenedores. Hay imágenes oficiales de por ejemplo Ubuntu, Apache, etc, que fueron creadas por sus creadores oficiales.

Página oficial para imágenes: <https://hub.docker.com>

Vamos a utilizar la siguiente imagen para pruebas:

https://hub.docker.com/_/hello-world

Para ejecutar este contenedor “hello-word” escribimos en la terminal:

```
1 docker run hello-world
```

Una vez ejecutado, ya dispondremos de la imagen descargada, podemos ver todas las imágenes que tenemos descargadas con:

```

1 docker images
2
3 REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
4 hello-world         latest   ee301c921b8a  9 months ago  9.14kB

```

Desde la página web de docker hub, podemos ver diferentes versiones de la misma imagen en la pestaña “TAGS”

TAG

latest
Last pushed 4 days ago by [doijankv](#)

DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
dbbd3cf66631	linux/386	None found	2.66 KB
245fe15fb8f	windows/amd64	None found	115.14 MB
088bdbea94d5	windows/amd64	None found	99.78 MB
+8 more...			

TAG

nanoserver-Itsc2022
Last pushed 4 days ago by [doijankv](#)

DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
245fe15fb8f	windows/amd64	None found	115.14 MB

Podemos descargar una imagen específica y ejecutarla:

```

1 docker run hello-world:linux
2
3 docker images
4 REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
5 hello-world        latest   ee301c921b8a  9 months ago  9.14kB
6 hello-world        linux    ee301c921b8a  9 months ago  9.14kB
7
8 docker run hello-world:linux

```

Para eliminar una imagen utilizamos el parámetro `rmi` por ejemplo:

```

1 docker pull alpine
2
3 docker images
4 REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
5 alpine              latest   ace17d5d883e  3 weeks ago  7.73MB
6 hello-world         latest   ee301c921b8a  9 months ago  9.14kB
7 hello-world         linux    ee301c921b8a  9 months ago  9.14kB
8
9 # Eliminamos, opción 1
10 docker rmi ace17d5d883e
11
12 # Eliminamos, opción 2
13 docker rmi alpine
14
15 # Eliminamos, opción 3
16 docker rmi ace1

```

También se pueden buscar imágenes desde consola.

```

1 docker search ubuntu
2
3 NAME                  DESCRIPTION           STARS     OFFICIAL
4 ubuntu                Ubuntu is a Debian-based Linux operating sys... 16888  [OK]
5 websphere-liberty     WebSphere Liberty multi-architecture images ... 298  [OK]
6 open-liberty          Open Liberty multi-architecture images based... 64  [OK]
7 neurodebian          NeuroDebian provides neuroscience research s... 106  [OK]
8 ubuntu-debootstrap    DEPRECATED; use "ubuntu" instead 52  [OK]
9 ubuntu-upstart        DEPRECATED, as is Upstart (find other proces... 115  [OK]
10 ubuntu/nginx         Nginx, a high-performance reverse proxy & we... 112
11 ubuntu/squid         Squid is a caching proxy for the Web. Long-t... 83
12 ubuntu/cortex        Cortex provides storage for Prometheus. Long...
13 ubuntu/prometheus    Prometheus is a systems and service monitori... 56
14 ubuntu/apache2        Apache, a secure & extensible open-source HT...
15 ...

```

Volúmenes

Los volúmenes sirven para almacenar información de manera persistente en uno o varios contenedores. Es útil para que los archivos ya estén integrados en el propio contenedor y podamos disponer de dichos archivos en diferentes contenedores diferentes.

También nos permiten compartir archivos con el contenedor. Modificarlos en local y que se modifiquen en el contenedor.

OPERACIONES CON VOLÚMENES

```

1 # Ver disponibles
2 docker volume ls
3 DRIVER      VOLUME NAME
4
5 # Creamos un volumen
6 docker volume create almacen
7 almacen
8
9 docker volume ls
10 DRIVER      VOLUME NAME
11 local      almacen
12
13 docker volume inspect almacen
14 [
15   {
16     "CreatedAt": "2024-02-20T11:10:58Z",
17     "Driver": "local",
18     "Labels": null,
19     "Mountpoint": "/var/lib/docker/volumes/almacen/_data",
20     "Name": "almacen",
21     "Options": null,
22     "Scope": "local"
23   }
24 ]
25
26 # Lo borramos
27 docker volume rm almacen

```

EJEMPLO: COMPARTIR VOLÚMENES CON HOST

```

1 # Creamos un punto de montaje
2 docker run --rm -it -v /tmp/puntomontaje:/home ubuntu

```

EJEMPLO: COMPARTIR VOLÚMENES CON CONTENEDORES

Vamos a crear un volumen para compartir archivos entre nuestro sistema de ficheros local y 2 contenedores (ubuntu y fedora).

```

1 docker volume create almacen
2
3 # Descargamos la imagen de ubuntu
4 docker pull ubuntu
5 # Descargamos la imagen de fedora
6 docker pull fedora
7
8 docker images
9 REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
10 ubuntu              latest   a50ab9f16797  7 days ago    69.2MB
11 fedora              latest   46243415778a  2 months ago  259MB
12
13 ## Creamos un contenedor, modo interactivo
14 docker run --rm -it -v almacen:/home ubuntu
15 # dentro del contenedor
16 cd /home
17 touch prueba.txt
18 exit # -> Salimos del contenedor
19
20 ## Creamos otro contenedor, modo interactivo
21 docker run --rm -it -v almacen:/home fedora
22 # dentro del contenedor
23 cd /home
24 ls -> existe el archivo prueba.txt

```

Diferencias docker-compose vs DockerFile

<https://blog.elhacker.net/2022/01/gestion-contenedores-dockerfile-y-docker-compose.html>

DOCKER COMPOSE

Docker Compose es una herramienta que permite simplificar el uso de Docker. A partir de archivos YAML es mas sencillo crear contenedores, conectarlos, habilitar puertos, volúmenes, etc.

Con Compose puedes crear diferentes contenedores y al mismo tiempo, en cada contenedor, diferentes servicios, unirlos a un volúmen común, iniciarlos y apagarlos, etc. Es un componente fundamental para poder construir aplicaciones y microservicios

Parámetros docker-compose.yml

- **“version ‘3’**: Los archivos docker-compose.yml son versionados, lo que significa que es muy importante indicar la versión de las instrucciones que queremos darle. A medida de que Docker evoluciona, habrá nuevas versiones, pero de todos modos, siempre hay compatibilidad hacia atrás, al indicar la versión
- **“build .”**: Se utiliza para indicar donde está el Dockerfile que queremos utilizar para crear el contenedor. Al definir “.” automáticamente considerará el Dockerfile existente en directorio actual.
- **“command”**: Una vez creado el contenedor, aquí lanzamos el comando que permite ejecutar Jekyll, en modo servidor. El comando “–host 0.0.0.0” sirve para mapear el contenedor al sistema operativo host
- **“ports”**: mapeamos los puertos locales, por ejemplo 4000 (webserver jekyll) y 35729 (livereload) al servidor host. Esto permite que accediendo a Localhost:4000 podamos probar el sitio generador por Jekyll
- **“volumes”**: lo que hacemos es mapear el directorio local se mapee directamente con el /directoriox, lugar donde hemos creado la aplicación. De este modo, cualquier cambio en el directorio local en el host, se hará de inmediato en el contenedor.



Ejemplo, creación contenedor con wordpress:

```

1  mkdir /tmp/wp
2  cd /tmp/wp
3
4  nano docker-compose.yml

```

```

1 version: '3' #Utilizamos la versión 3
2
3 ## Nos saltamos la sección network
4
5 services:
6   db:
7     image: mariadb:10.3.9
8     volumes:
9       - data:/var/lib/mysql
10    environment:
11      - MYSQL_ROOT_PASSWORD=secret
12      - MYSQL_DATABASE=wordpress
13      - MYSQL_USER=manager
14      - MYSQL_PASSWORD=secret
15    ## Equivalente a
16    ## docker run -d --name wordpress-db \
17    ## --mount source=wordpress-db,target=/var/lib/mysql \
18    ## -e MYSQL_ROOT_PASSWORD=secret \
19    ## -e MYSQL_DATABASE=wordpress \
20    ## -e MYSQL_USER=manager \
21    ## -e MYSQL_PASSWORD=secret mariadb:10.3.9
22
23 web:
24   image: wordpress:4.9.8
25   depends_on:
26     - db
27   volumes:
28     - ./target:/var/www/html
29   environment:
30     - WORDPRESS_DB_USER=manager
31     - WORDPRESS_DB_PASSWORD=secret
32     - WORDPRESS_DB_HOST=db
33   ports:
34     - 8080:80
35     ## Equivalente a
36     ## docker run -d --name wordpress \
37     ## --link wordpress-db:mysql \
38     ## --mount type=bind,source="$(pwd)"/target,target=/var/www/html \
39     ## -e WORDPRESS_DB_USER=manager \
40     ## -e WORDPRESS_DB_PASSWORD=secret \
41     ## -p 8080:80 \
42     ## wordpress:4.9.8
43
44 volumes:
45   data: # creación de volumen, compose añade un prefijo por lo que se llamará worpdress_data

```

```

1 # Levantamos la aplicación
2 docker-compose up -d
3 # El parámetro -d es similar al de docker run: nos permite levantar los servicios en segundo plano.
4
5 docker-compose ps
6 ## docker-compose ps solo muestra información de los servicios que se define en docker-compose.yaml, mientras que docker muestra todos.
7
8 # Detener servicios
9 docker-compose stop
10
11 # Borrar
12 docker-compose down
13
14 # Borrar volúmenes
15 docker-compose down -v

```

Cuando creamos contenedores con `docker` sin indicar un nombre, por defecto asigna uno aleatorio; mientras que en *Compose* el prefijo es el nombre del directorio y el sufijo el nombre del servicio: **wordpress_db_1**. El número indica el número de instancia. Es posible levantar más de una instancia de un mismo servicio.

Equivalencia de parámetros

parámetro <i>Docker</i>	parámetro <i>Composer</i>
<code>--mount</code>	<code>volumes</code>
<code>-e</code>	<code>environment</code>
<code>-p, --publish</code>	<code>ports</code>
se escribe el nombre de la imagen	<code>image</code>

Si reiniciamos el ordenador, los contenedores estarán detenidos (`stop`), podremos reiniciarlos con `docker start` o `docker-compose start`. Este es el comportamiento predeterminado y el que nos interesa en un entorno de desarrollo.

Sin embargo, en otros entornos, o para casos concretos, igual queremos que un contenedor tenga el mismo estado en el que estaba antes de reiniciar la máquina (iniciado o parado).

Para eso usaremos el parámetro `restart`. En el caso de la base de datos de nuestro ejemplo, la configuración quedaría como:

```

1 services:
2   db:
3     image: mariadb:10.3.9
4     restart: unless-stopped
5     volumes:
6       - data:/var/lib/mysql
7     environment:
8       - MYSQL_ROOT_PASSWORD=secret
9       - MYSQL_DATABASE=wordpress
10      - MYSQL_USER=manager
11      - MYSQL_PASSWORD=secret

```

Otros valores son: `no` (por defecto), `always` y `on-failure`.

`DOCKERFILE`

Ejemplo básico

Nos ubicamos en la carpeta donde vayamos a trabajar con la imagen, por ejemplo voy a crear un directorio llamado `docker-images`.

Y creamos una archivo `Dockerfile` con un editor de texto.

```

1 FROM ubuntu
2 RUN apt update
3 RUN apt install python 3 -y
4 RUN apt install netris -y

```

Creamos la imagen según las órdenes anteriores:

```
1 docker build -t python-ubuntu .
```

Hay un error en el Dockerfile, corregir y volver a ejecutar el build.

Ya tenemos una imagen de ubuntu pero con python3 instalado:

```

1 docker run -it python-ubuntu
2
3 python3
4
5 Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
6 Type "help", "copyright", "credits" or "license" for more information.
7 >>>

```

Una vez salimos podemos ver todas las “capas” de la imagen:

```

1 docker history -H python-ubuntu
2
3 IMAGE      CREATED      CREATED BY
4 78732fd773c4  2 minutes ago  RUN /bin/sh -c apt install netris # buildkit      1MB      buildkit.dockerfile.v0
5 <missing>    2 minutes ago  RUN /bin/sh -c apt install python3 -y # buil...  29.5MB   buildkit.dockerfile.v0
6 <missing>    3 minutes ago  RUN /bin/sh -c apt update # buildkit      45.1MB   buildkit.dockerfile.v0
7 <missing>    3 weeks ago   /bin/sh -c #(nop)  CMD ["#!/bin/bash"]
8 <missing>    3 weeks ago   /bin/sh -c #(nop) ADD file:8d91b8bd386e0cc34...  69.2MB
9 <missing>    3 weeks ago   /bin/sh -c #(nop) LABEL org.opencontainers...
10 <missing>   3 weeks ago   /bin/sh -c #(nop) LABEL org.opencontainers...
11 <missing>   3 weeks ago   /bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH  0B
12 <missing>   3 weeks ago   /bin/sh -c #(nop) ARG RELEASE        0B

```

Creación de imágenes propias

Para construir una imagen, se crea un `Dockerfile` con las instrucciones que especifican lo que va a ir en el entorno, dentro del contenedor (redes, volúmenes, puertos al exterior, archivos que se incluyen).

- Indica cómo y con qué construir la imagen.
- Podemos utilizar la imagen en tantos contenedores como queramos.



El `Dockerfile` nos permitirá definir las funciones básicas del contenedor.

Todo `Dockerfile` debe terminar en un comando `CMD` o en un `ENTRYPOINT`, pero en este caso, no lo utilizamos, ya que lanzaremos un comando directamente desde la receta de DockerCompose. Es decir, este `Dockerfile` se utiliza solamente para construir el contenedor y configurarlo. No es auto-ejecutable.

Construir la imagen desde el Dockerfile

```
1 docker build -t nombre-de-tu-imagen .
```

Ejemplo:

```
1 docker build -t mi-aplicacion:1.0 .
```

Parámetros importantes:

- `-t` : Etiqueta (tag) para tu imagen
- `.` : Ruta donde está el Dockerfile (usualmente el directorio actual)

Crear y ejecutar el contenedor

```
1 docker run -d --name nombre-contenedor nombre-de-tu-imagen
```

Ejemplo:

```
1 docker run -d --name mi-contenedor mi-aplicacion:1.0
```

Opciones

FROM

Imagen del sistema operativo donde va a correr el contenedor.

Consejo

Las versiones "Alpine linux" ocupan muy poco espacio.

RUN

El comando `RUN` se ejecuta cuando se está construyendo una imagen personalizada para realizar una acción, creando una capa nueva. Este comando tiene el siguiente formato:

```
1 RUN comando
```

```
1 RUN ["ejecutable", "parametro1", ...]
```

Ejemplo en windows:

```
1 RUN ["Powershell", "Get-Services", "*"]
```

COPY

Sirve para copiar archivos desde nuestra máquina al contenedor. Podemos pasar un documento de texto de la máquina anfitrión al contenedor de python-ubuntu.

```
1 FROM ....
2 RUN ....
3 RUN ....
4 COPY prueba.txt /
```

Volvemos a construir la imagen y accedemos a ella para buscar el archivo.

ENV

Podemos crear una variable y enviarla a nuestro contenedor, en mi caso por ejemplo voy a definir una variable llamada contenido que va a ir dirigida a un bloc de notas que está dentro del contenedor:

```
1 ....
2 ENV NUEVO_PATH /etc
```

Una vez regenerado la imagen y dentro del contenedor:

```
1 echo $NUEVO_PATH
2 /etc
```

Podemos combinar RUN con ENV

```
1 ENV NUEVO_PATH /etc
2 RUN echo $NUEVO_PATH > /prueba.txt
```

WORKDIR

Nos situamos en un directorio determinado, nos puede ayudar en la copia de ficheros.

```
1 ....
2 WORKDIR /home
3 COPY prueba.txt . # Lo copia en /home
```

EXPOSE

Permite exponer los puertos que queramos

LABEL

Creamos etiquetas, por ejemplo:

```
1 FROM ubuntu
2 LABEL version=1.0
3 LABEL autor=JosepGarcia
4 ....
```

USER

Sirve para establecer el usuario, debe existir. (Por defecto se utiliza root).

```
1 ....
2 RUN echo $(whoami) > /tmp/usuarioantes.txt
3
4 RUN useradd -m josepgarcia
5 USER josepgarcia
6 WORKDIR /home/josepgarcia
7 RUN echo $(whoami) > /tmp/usuarioahora.txt
```

CMD

Ejecuta comandos una vez se ha inicializado el contenedor (RUN se utiliza para crear la imagen de un contenedor).

```
1 ## Ejecutamos el comando top cuando se inicie el contenedor
2 ....
3 CMD top
```

IGNORE

Sirve para ignorar aquello que tengamos en nuestro directorio actual.

Por ejemplo:

Creamos una imagen que copie todo nuestro directorio actual al contenedor.

```
1 ls
2 Dockerfile prueba.txt
```

contenido de Dockerfile:

```
1 ...
2 COPY . /tmp
3 ...
```

Ahora le decimos que copie todo menos el archivo `Dockerfile`, para ello creamos un fichero llamado `.dockerignore` con el siguiente contenido

```
1 Dockerfile
```

Guardar estado de los contenedores

<https://www.baeldung.com/ops/docker-save-container-state>

Casos de uso**Compatibilidad de código entre diferentes versiones de un lenguaje**

```
1 mkdir /tmp/php
2 cd /tmp/php
```

Crear el siguiente archivo (`test.php`)

```
1 <?php
2 // Funciona bien en PHP5 ya que list hace la asignación desde el último al primero
3 // En PHP 5, list() asigna los valores empezando desde el parámetro más a la derecha. En PHP 7, list() empieza desde el parámetro más a la izquierda.
4 // https://www.php.net/manual/es/function.list.php
5 $info = array('cafeína', 'marrón', 'café');
6
7 // Enumerar todas las variables
8 list($datos[], $datos[], $datos[]) = $info;
9 echo "El $datos[0] es $datos[1] y la $datos[2] lo hace especial.\n";
10 ?>
```

A continuación vamos a crear dos contenedores que sirva este código usando imágenes distintas , para cada versión de PHP y usando puertos distintos para acceder a cada versión de la aplicación:

```

1 $ docker run -d -p 8081:80 --name php56 -v /tmp/php:/var/www/html:ro php:5.6-apache
2 Unable to find image 'php:5.6-apache' locally
3 5.6-apache: Pulling from library/php
4 5e6ec7f28fb7: Pull complete
5 cf165947b5b7: Pull complete
6 7bd37682846d: Pull complete
7 99daf8e838e1: Pull complete
8 ae320713efba: Pull complete
9 ebc99c48d8c: Pull complete
10 9867e71b4ab6: Pull complete
11 936eb418164a: Pull complete
12 bc298e7adaf7: Pull complete
13 ccd61b587bcd: Pull complete
14 b2d4b347f67c: Pull complete
15 56e9dde34152: Pull complete
16 9ad99b17eb78: Pull complete
17 Digest: sha256:0a40fd273961b99d8afe69a61a68c73c04bc0caa9de384d3b2dd9e7986eec86d
18 Status: Downloaded newer image for php:5.6-apache
19 10c4c965e067c93cf0ff261ddbf358e323226ed3d54df3b5f4e69c64615194fd
20
21 #Buscamos el id del contenedor que acabamos de crear
22 $ docker ps
23 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
24 10c4c965e067 php:5.6-apache "docker-php-entrypoi..." 18 seconds ago Up 17 seconds 0.0.0.0:8081->80/tcp, :::8081->80/tcp php56
25
26 #Accedemos a su consola con el id que hemos encontrado
27 $ docker exec -it 10c4c965e067 /bin/bash
28
29 #Ejecutamos test.php y vemos que funciona perfectamente
30 root@10c4c965e067:/var/www/html# php test.php
31 El café es marrón y la cafeína lo hace especial.
32
33 #Salimos del contenedor
34 root@10c4c965e067:/var/www/html# exit

```

Ahora lo intentaremos con una versión más reciente de php, la 7.4:

```

1 $ docker run -d -p 8082:80 --name php74 -v /tmp/php:/var/www/html:ro php:7.4-apache
2 Unable to find image 'php:7.4-apache' locally
3 7.4-apache: Pulling from library/php
4 a603fa5e3b41: Pull complete
5 c428f1a49423: Pull complete
6 156740b07ef8: Pull complete
7 fb5a4cb8af82f: Pull complete
8 25f85b498f65: Pull complete
9 9b233e420ac7: Pull complete
10 fe42347c4ecf: Pull complete
11 d14eb2ed1e17: Pull complete
12 66d98f73acb6: Pull complete
13 d2c43c5efbcb: Pull complete
14 ab590b48ea47: Pull complete
15 80692ae2d067: Pull complete
16 05e465aaa99a: Pull complete
17 Digest: sha256:c9d7e608f73832673479770d66aac8100011ec751d1905ff63fae3fe2e0ca6d
18 Status: Downloaded newer image for php:7.4-apache
19 de1a3c9f72a4003022a0dfa9e1cdd7d2733ae52d21046e342370131abc50a112
20
21 #Ahora ejecutaremos test.php de otro modo, directamente accediendo al servidor web que hemos levantado en el puerto 8082 accediendo a la url http://localhost:8082/test.php

```

El resultado desde el navegador debería ser similar a:



El café es marrón y la café lo hace especial.

Crear una imagen de un repositorio de github

Ejemplo repositorio:

<https://github.com/k4m4/kickthemout>

Creamos el Dockerfile :

```

1 FROM ubuntu:focal
2
3 RUN apt update -y && apt upgrade -y && apt install python3 -y
4 RUN apt install -y git
5 RUN apt install -y python3-pip
6 RUN apt install -y nmap
7 RUN git clone https://github.com/k4m4/kickthemout.git
8
9 WORKDIR /kickthemout
10
11 RUN pip3 install -r requirements.txt
12
13 CMD python3 kickthemout.py

```

Si ahora creamos nuestra imagen a partir del Dockerfile:

```

1 $ docker build -t kick:1.0 .
2 [+] Building 111.2s (11/11) FINISHED                                            docker:desktop-linux
3 => [internal] load build definition from Dockerfile                           0.1s
4 => => transferring dockerfile: 316B                                         0.0s
5 => [internal] load metadata for docker.io/library/ubuntu:focal                2.0s
6 => [internal] load .dockerignore                                              0.0s
7 => => transferring context: 2B                                              0.0s
8 => [1/7] FROM docker.io/library/ubuntu:focal@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2faf 2.0s
9 => => resolve docker.io/library/ubuntu:focal@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2faf 0.0s
10 => => sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09c2476 27.51MB / 27.51MB   1.2s
11 => => extracting sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09c2476 0.7s
12 => [2/7] RUN apt update -y && apt upgrade -y && apt install python3 -y          30.5s
13 => [3/7] RUN apt install -y git                                               28.7s
14 => [4/7] RUN apt install -y python3-pip                                       33.2s
15 => [5/7] RUN git clone https://github.com/k4m4/kickthemout.git               1.0s
16 => [6/7] WORKDIR /kickthemout                                              0.1s
17 => [7/7] RUN pip3 install -r requirements.txt                                3.2s
18 => exporting to image                                                       10.0s
19 => => exporting layers                                                       6.8s
20 => => exporting manifest sha256:771f59312051c037aaff14313c8b0edf1e4783c1b35b62015216ef06cd7b4270 0.0s
21 => => exporting config sha256:8a78e3ef64feef584340963a82a193bc3949433f74fa5863cf3e874cc6cd95025 0.0s
22 => => exporting attestation manifest sha256:98f1e53e84cf3c2fa9f69b4dd88d852ca0524e060a268f82ed3dadc386bd655 0.1s
23 => => exporting manifest list sha256:36321bf83b1c51a1200aa0742bd671f14d999633640535cfb2c72ad530c25c48 0.0s
24 => => naming to docker.io/library/kick:1.0                                 0.0s
25 => => unpacking to docker.io/library/kick:1.0                            2.9s
26
27 1 warning found (use docker --debug to expand):
28 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 12)
29
30 View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ck1d95jji381mhtqspu1azv13

```

A continuación podremos crear un nuevo contenedor usando esa imagen:

```

1 $ docker run -it --name mi-kick kick:1.0
2
3 ERROR: Gateway IP could not be obtained. Please enter IP manually.
4
5 kickthemout> Enter Gateway IP (e.g. 192.168.1.1):

```

Vemos que el script no encuentra la puerta de enlace, pero la imagen y el contenedor creados funcionan correctamente.

Dockerizar nuestro jupyter con carpeta para notebooks

Otra forma de levantar un contenedor con docker-compose es utilizar un Dockerfile para generar la imagen (en lugar de usar una de DockerHub)

Necesitamos una carpeta con la siguiente estructura:

```

1 .
2   └── docker-compose.yml
3   ├── Dockerfile
4   └── notebooks
5     └── rockpaperscissors.ipynb

```

El fichero Dockerfile tiene el siguiente contenido:

```

1 #versión de python compatible con experta
2 FROM python:3.6
3 #librería de python para sistemas expertos
4 RUN pip3 install experta
5 #librería para usar notebooks en python
6 RUN pip3 install notebook
7 #creamos la carpeta de trabajo
8 WORKDIR /home/MIA2526
9 #ejecutamos el jupyter notebook en el puerto 8888
10 CMD jupyter notebook --allow-root --ip=0.0.0.0 --port=8888 --no-browser

```

Ahora, para el docker-compose.yml tendremos:

```

1 services:
2   experta:
3     container_name: experta #bautizamos a nuestro contenedor
4     build: . #con esta linea le indicamos que busque el Dockerfile, en lugar de buscar una imagen en un repositorio
5     ports:
6       - "8888:8888" #publicamos el puerto para que sea accesible desde fuera
7     volumes:
8       - ./notebooks:/home/MIA2526 #aquí asignamos la carpeta local notebooks con la carpeta de trabajo del contenedor

```

Y por último necesitamos el notebook para hacer pruebas, en nuestro caso es un juego de piedra, papel o tijeras:

[rockpaperscissors.ipynb](#) este fichero debemos guardarla en la carpeta notebooks

Ahora con toda la estructura lista y desde la raíz de la carpeta (donde estan el docker-compose.yml y el Dockerfile) ejecutamos:

```

1 $ docker-compose up
2 [+]
3   ✓ Container experta Recreated
4 Attaching to experta
5 experta | [I 16:30:08.568 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
6 experta | [I 16:30:08.784 NotebookApp] Serving notebooks from local directory: /home/MIA2324
7 experta | [I 16:30:08.784 NotebookApp] Jupyter Notebook 6.4.10 is running at:
8 experta | [I 16:30:08.784 NotebookApp] http://e0ec65ac1d84:8888/?token=825b1ba76502787821bc045496e3429bca02ba09720a835b
9 experta | [I 16:30:08.784 NotebookApp] or http://127.0.0.1:8888/?token=825b1ba76502787821bc045496e3429bca02ba09720a835b
10 experta | [I 16:30:08.784 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
11 experta | [C 16:30:08.787 NotebookApp]
12 experta |
13 experta | To access the notebook, open this file in a browser:
14 experta | file:///root/.local/share/jupyter/runtime/nbserver-7-open.html
15 experta | Or copy and paste one of these URLs:
16 experta | http://e0ec65ac1d84:8888/?token=825b1ba76502787821bc045496e3429bca02ba09720a835b
17 experta | or http://127.0.0.1:8888/?token=825b1ba76502787821bc045496e3429bca02ba09720a835b

```

Ahora podemos hacer click directamente sobre el enlace a <http://127.0.0.1:8888/?token=bebf660273e8e168c7fec90978ed56fb50db6b08d915cb14> donde veremos nuestro jupyter notebook:

The screenshot shows a browser window with the Jupyter logo in the top left. The main content area displays a file list with two items: '0' and 'rockpaperscissors.ipynb'. At the top right, there are buttons for 'Quit', 'Logout', 'Upload', 'New', and a refresh icon. Below the file list, there are buttons for 'Name', 'Last Modified', and 'File size'.

Name	Last Modified	File size
0	fa 2 anys	10 kB
rockpaperscissors.ipynb		

Si hacemos click sobre el notebook `rockpaperscissors.ipynb` podremos ver:



```
In [1]: from experta import *
from experta.fact import *
import random

NERD = True

In [2]: class WinTotals(Fact):
    human = Field(int, default=0)
    computer = Field(int, default=0)
    ties = Field(int, default=0)

    class Results(Fact):
        winner = Field(str, mandatory=True)
        loser = Field(str, mandatory=True)
        why = Field(str, mandatory=True)

    class ValidAnswer(Fact):
        answer = Field(str, mandatory=True)
        key = Field(str, mandatory=True)

    class Action(Fact):
        pass

class HumanChoice(Fact):
    ...

```

Después de pulsar varias veces (para ir ejecutando todas las celdas) podremos ver como evoluciona nuestro juego:

```
In [*]: rps.reset()
rps.run()
```

```
Lets play a game!
You choose rock, paper, or scissors,
and I'll do the same.
Scissors (S), Paper (P), Rock (R)? R
Computer wins! Paper covers rock
Play again?Y
Scissors (S), Paper (P), Rock (R)? S
Tie! Ha-ha!
Play again?Y
Scissors (S), Paper (P), Rock (R)? P
You win! Paper covers rock
```

```
Play again? |
```

Para detener el contenedor que hemos lanzado con `docker-compose`, solo hemos de pulsar `^ Ctrl + C`

Crear una imagen personalizada

<https://jolthgs.wordpress.com/2019/09/25/create-a-debian-container-in-docker-for-development/>

Para crear una imagen personalizada utilizaremos el contenedor que habíamos creado en el [primer ejemplo](#), con una debian actualizada y con un fichero de texto `prueba.txt`

```

1 # Iniciamos el contenedor
2 $ docker start debian-mini
3
4 $ docker ps
5 CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
6 8c6b29c818ee   debian:13-slim   "/bin/bash -l"   About a minute ago   Up 6 seconds
7
8 # Entramos en el contenedor iniciado
9 $ docker exec -it debian-mini bash
10
11 # Instalamos el paquete
12 apt install bsdgames
13
14 # Ver todos los juegos incluidos
15 dpkg -L bsdgames | grep /usr/games
16
17 # Ejecutamos el tetris (por ejemplo)
18 /usr/games/tetris-bsd
19 # Salimos del juego con la q
20
21 # Salimos del contenedor
22 control + d
23
24 # Nuestra imagen
25 $ docker image ls
26 REPOSITORY   TAG      IMAGE ID      CREATED      SIZE
27 debian       13-slim  1caf1c703c8f  42 hours ago  117MB

```



Copias de seguridad

Copias de contenedores

Ya estén encendidos o apagados, podemos realizar respaldos de seguridad de los contenedores. Utilizando la opción “*export*” empaquetará el contenido, generando un fichero con extensión “*.tar*” de la siguiente manera:

```
1 docker export -o fichero-resultante.tar nombre-contenedor
```

0

```
1 docker export nombre-contenedor > fichero-resultante.tar
```

Restauración de copias de seguridad de contenedores

Hay que tener en cuenta, antes de nada, que no es posible restaurar el contenedor directamente, de forma automática. En cambio, sí podemos crear una imagen, a partir de un respaldo de un contenedor, mediante el parámetro “*import*” de la siguiente manera:

```
1 docker import fichero-backup.tar nombre-nueva-imagen
```

Copias de imágenes

Aunque no tiene mucho sentido por que se bajan muy rápido, también tenemos la posibilidad de realizar copias de seguridad de imágenes. El proceso se realiza al utilizar el parámetro 'save', que empaquetará el contenido y generará un fichero con extensión "tar", así:

```
1 docker save nombre_imagen > imagen.tar
```

0

```
1 docker save -o imagen.tar nombre_imagen
```

Restaurar copias de seguridad de imágenes

Con el parámetro 'load', podemos restaurar copias de seguridad en formato '.tar' y de esta manera recuperar la imagen.

```
1 docker load -i fichero.tar
```

Contenedores ejemplo

Monitorización y Gestión

- **Netdata** - Monitorización de sistemas en tiempo real con dashboard web completo
- **Glances** - Monitorización del sistema via web que muestra información de CPU, memoria, red y procesos
- **Portainer** - Interfaz web para gestionar y administrar contenedores Docker
- **Uptime Kuma** - Monitor de disponibilidad para verificar el estado de contenedores y servicios

Proxy y Red

- **Nginx Proxy Manager** - Proxy inverso con interfaz web para gestionar hosts virtuales y certificados SSL/TLS
- **Acestream** - Motor P2P para streaming de video, útil para integrar canales en Jellyfin
- **Libreddit** - Interfaz alternativa para Reddit libre de publicidad y tracking (modo lectura)
- **Invidious** - Interfaz alternativa para YouTube que respeta la privacidad

Multimedia y Entretenimiento

- **Jellyfin** - Servidor multimedia libre para organizar y streaming de películas, series y música
- **Soulseek** - Cliente para la red P2P Soulseek especializada en compartir música
- **youtube-dl** - Herramienta para descargar videos y audio de YouTube y otros sitios
- **Photoprism** - Servicio de gestión y visualización de fotos personales con funciones de IA

Productividad y Organización

- **Nextcloud** - Plataforma de colaboración y almacenamiento en la nube auto-alojado
- **Linkding** - Marcador social para guardar y organizar enlaces web
- **GitBucket** - Plataforma Git auto-alojada similar a GitHub
- **SearXNG** - Metabuscador privado que agrega resultados de múltiples motores de búsqueda

Seguridad y Contraseñas

- **Vaultwarden** - Implementación alternativa del gestor de contraseñas Bitwarden, más ligera y eficiente

Automatización del Hogar

- **Home Assistant** - Plataforma de domótica de código abierto para automatizar y controlar dispositivos del hogar
- **Homebridge** - Puente que permite integrar dispositivos no compatibles con el ecosistema Apple HomeKit
- **Camera.UI** - Aplicación para gestionar y visualizar cámaras de seguridad con integración HomeKit

Utilidades y Mantenimiento

- **Homepage** - Dashboard personalizable como página de inicio para acceder a todos los servicios
- **Watchtower** - Servicio que actualiza automáticamente los contenedores Docker cuando hay nuevas versiones disponibles

Ejercicios

Ejercicio1

Instala docker desktop en tu PC. Guarda una captura de pantalla que demuestre que funciona en tu PC.

Ejercicio2

Asegúrate de que puedes reproducir todos los casos de uso vistos [más arriba](#). Guarda varias capturas de pantalla donde se pueda ver que han funcionado los diferentes casos.

Ejercicio 3

Crea 2 contenedores, uno con python 3.11 y otro con python 3.9. Explica los pasos que has seguido para conseguirlo y muestra pantalla con los contenedores funcionando.

Ejercicio 4

Elige alguna de las imágenes propuestas en los [Contenedores de ejemplo](#) (o cualquiera a tu elección de [DockerHub](#)), escribe un `docker-compose.yml` y muéstralos con capturas y en funcionamiento en la memoria en PDF.

Tarea entregable

Recopila todas las capturas y explicaciones de los 4 ejercicios en una memoria en PDF que justifique todo el trabajo realizado.

 12 de noviembre de 2025

3. UD01

3.1 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:

Definición

"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:

Definición

"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.

Recuerda

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 --> A
    A -- Responde --> C
    C --> 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



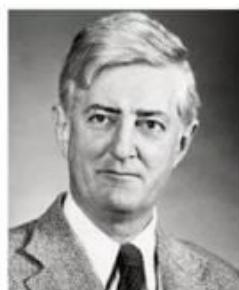
Alan Newell



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 Kasparov contra la máquina. La primera partida la ganó Deep Blue, otras tres las ganó Kasparov 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.

Curiosidad

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 vídeo de los 500 millones de dólares". En dicho vídeo 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 vídeo).



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 **Tensor Flow**, 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.



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.

Curiosidad

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.

Curiosidad

¿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.
- 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.
- 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.

Importante

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.

Definición

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.

Definición

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.

Definición

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.

Ejemplo

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.

- 2. 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.
- 3. 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.
- 4. 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.

Importante

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

- 5. 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:

- 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.
- 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.
- 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.
- 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.

Ejemplo:**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 existe 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 ir 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)

Definición

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 suplirla 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

Definición

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

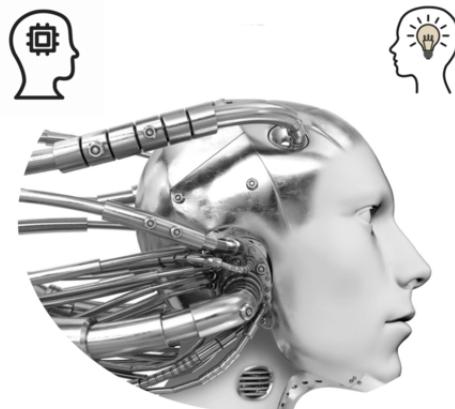
En noviembre de **2016**, **Arend Hintze**, profesor de la Universidad de Michigan e investigador en el campo de la Inteligencia Artificial, escribió un artículo titulado: "**Understanding the four types of AI, from reactive robots to self-aware**

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)

MEMORIA LIMITADA:

Máquinas que pueden almacenar conocimiento y usarlo para aprender y capacitarse para tareas futuras..



TEORÍA DE LA MENTE:

Concepto de IA que puede sentir y responder a las emociones humanas, así como realizar las tareas de las máquinas de memoria limitada.

MÁQUINAS REACTIVAS:

Tecnología capaz de responder a estímulos externos en tiempo real, pero incapaz de construir una base de memoria y almacenar información para uso futuro.



Néstor Márquez
Arend Hintze (Concepto)
Eric del Castillo
(Imagen de Homo Singularis).
@Creative Commons

CONSCIENTE DE SÍ MISMO:

Etapa final de la IA en la que las máquinas no solo pueden reconocer las emociones de los demás, sino que también tendrían un sentido de sí mismos y una inteligencia a nivel humano..



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:

```
graph LR
    D2["D2<br>(Datos y<br>Conocimiento)"] --> RBS["RBS<br>((Sistema Basado<br>en Reglas))"]
    RBS --> A["A<br>|Acción<br>Tomada| A<br>((Acción o<br>Decisión))"]
    A --> D2
```

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.

Curiosidad

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 **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 realizara por sí misma.

Ejemplo:

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:

Definición

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.**

Ejemplo:

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.

Importante

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:

Definición

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:

Definición

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

Ejemplo

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:

Definición

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.



Redes Neuronales Artificiales (SAA y PIA)

CONCEPTOS BÁSICOS Y FUNCIONAMIENTO

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

Definición

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

Definición

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.

Curiosidad

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.

Ejemplo

Aplicación de Visión por Computadora:

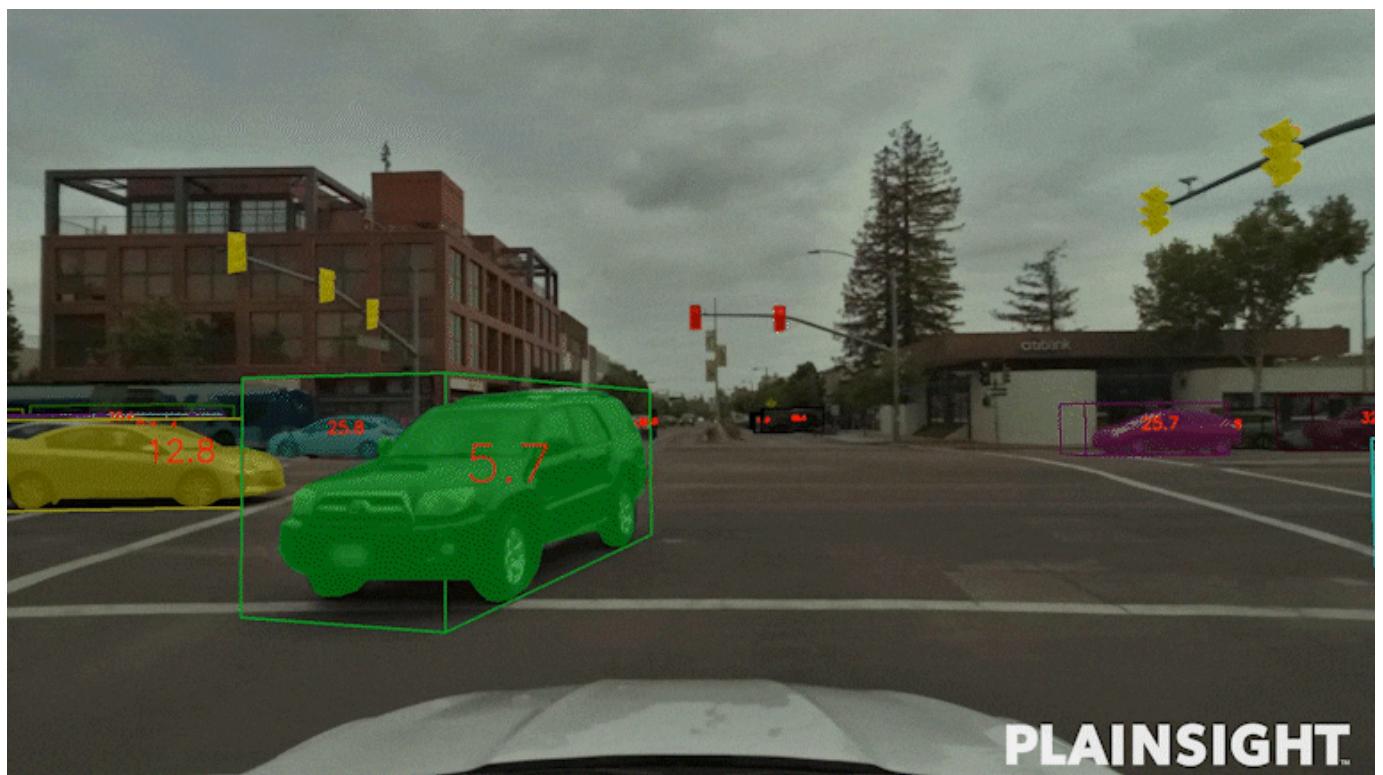
Un ejemplo práctico de aplicación de visión por computadora es el reconocimiento facial utilizado en aplicaciones de seguridad y desbloqueo de dispositivos. En este caso:

- **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.



PLAINSIGHT.

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.

Curiosidad

Si quieres 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.

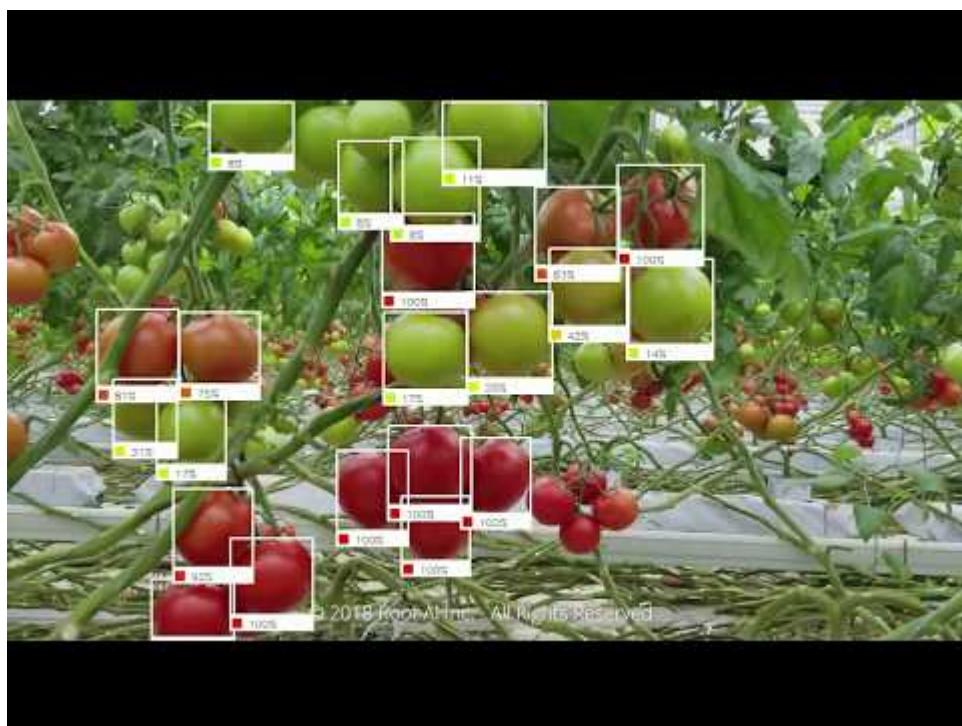
<https://bostondynamics.com/videos/>

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.



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 AI/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, eliminando 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

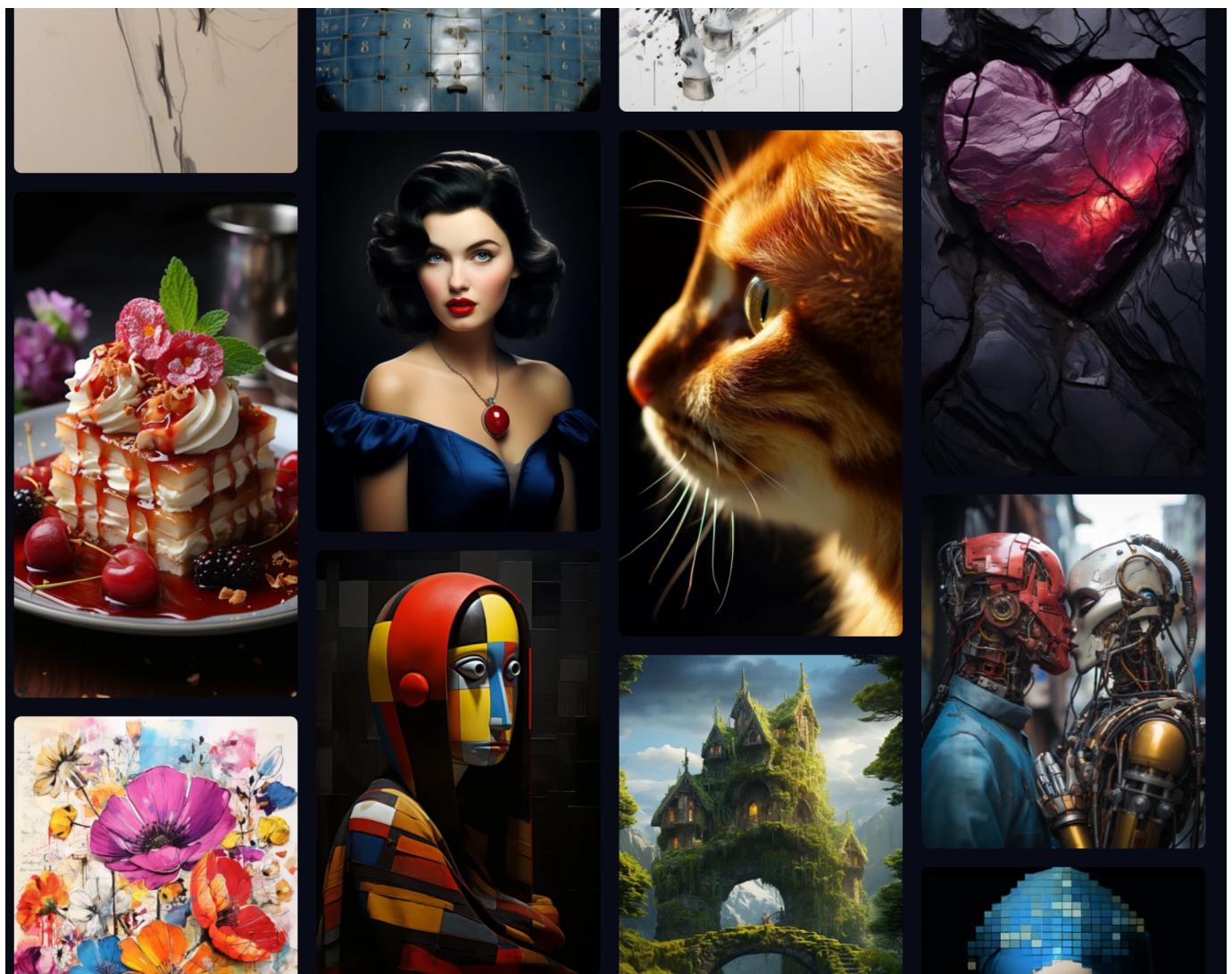
Uno de los grandes e inesperados avances de la computación de la década pasada, ha sido el de los modelos generativos: las redes GAN para el campo de la imagen y los modelos de generación de texto basados en Transformers.

- **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 aleatorio 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.



Curiosidad

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?

Ejemplos

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.

Ejemplo

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. **Ánalisis 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

Definición

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)

Definición

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

Definición

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</br>]
        (Definición)
        (Historia)
        (Futuro)
        (Sistemas</br>Inteligentes)
    [<span style="color: orange;>💡Clasificaciones</span>]
        (Según tareas)
        (Escuelas)
        (Russell/Norvig)
        (Hintze)
    [<span style="color: purple;>💡Utilización</span>Modelos]
        (Requisitos</br>de un SRP)
        (Modelos de</br>sistemas de IA)
    [<span style="color: green;>💡Técnicas</span>]
        (Sistemas</br>expertos)
        (Machine</br>Learning)
        (Redes</br>Neuronales)
        (Algoritmos</br>Genéticos)
        (Lógica</br>Difusa)
    [<span style="color: red;>💡Aplicaciones</span>]
        (Visión por</br>Computadora)
        (PLN)
)

```

(Analítica</br>avanzada)
(Robótica</br>e IA)
(Ciencia de datos</br>y Data Mining)
(Ciberseguridad)
[ Interacción]
(Voz)
(Cerebro</br>Computadora)
(RA y RV)

 7 de octubre de 2025

3.2 Diapositivas de la UD01

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

⌚17 de septiembre de 2025

3.3 Actividades

Robocode TankRoyale

Preparación del entorno

- Al menos java 11 (Yo estoy usando Java 23 (OpenJDK) y la versión 0.34.0 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.

Consejo

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

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 [este paquete](#) y abre el proyecto con el IDE IntelliJ.

Sin maven

Si no quieres usar maven y deseas montar el proyecto por tu cuenta sigue estas indicaciones:

También necesitaras la librería (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

```

Atención

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.

¡Jo!

Gracias a este fragmento, tu bot funcionará correctamente desde IntelliJ y desde el gui de RoboCode:

```

1 // Constructor, que carga el fichero de configuración
2 IABDBot() {
3   super(BotInfo.fromFile(getConfigPath()));
4 }
5
6 // Este método obtiene la ruta del fichero de configuración y se asegura que funcione desde IntelliJ y desde la gui de Robocode
7 private static String getConfigPath() {
8   String miBot = "IABDBot";
9   String configPath = "src/main/java/" + miBot + ".json";
10  java.io.File configFile = new java.io.File(configPath);
11  if (!configFile.exists()) {
12    configPath = miBot + ".json";
13  }
14  return configPath;
15 }

```

ÚLTIMAS NOVEDADES DEL PROYECTO

- Se pueden grabar los combates en un fichero dentro de la carpeta `recordings` con la extensión `battle.gz` que luego podemos reproducir en diferido. Esto nos permite revisar situaciones y estudiar estrategias de mejora.

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 **CEIABDEPM2025**

```

1 bots-secrets=CEIABDEPM2025
2 [...]

```

¡Jo!

Con las últimas versiones de la gui el uso de contraseñas en el servidor está inicialmente deshabilitado, si queremos habilitarlo lo podemos hacer desde el propio interfaz: `Config/Server Options/Enable server secrets` o bien desde el fichero de configuración `server.properties` añadiendo:

```
1 server-secrets-enabled=true
```

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=CEIABDEPM2025;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.

Atención

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
```

Consejo

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 verde = Color.fromRgb(0,255,0);
2  setBodyColor(Color.KHAKI);
3  setTurretColor(Color.KHAKI);
4  setRadarColor(Color.KHAKI);
5  setScanColor(Color.KHAKI);
6  setBulletColor(Color.KHAKI);

```

Puedes encontrar la lista completa de colores disponibles en la API de Robocode TankRoyale.

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     ...
12     if (getEnemyCount() > 10) {
13         // Un gran número de enemigos sugiere movimientos fluidos
14         modoRobot=MODO_RONDAR;
15     } else if (getEnemyCount() > 1) {
16         // esquivar es la mejor táctica para grupos pequeños
17         modoRobot=MODO_ESQUIVAR;
18     } else if (getEnemyCount() == 1) {
19         // Si solo queda un robot, persigue
20         modoRobot=MODO_ASESINO;
21     }
22     ...
23 }
24
25 public void doMove() {
26     switch (modoRobot){
27         case MODO_RONDAR:
28             //ronda
29             break;
30         case MODO_ESQUIVAR:
31             //esquiva
32             break;
33         case MODO_ASESINO:
34             //asesina
35             break;
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 cañon

```

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.

⌚19 de octubre 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](#)

 **Importante**

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.

2. 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.

3. ¿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.

4. 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 0.34.0 de la API** (Cambiado el 20/10/25, ya tenemos la versión multi-idioma y con escalado. A no ser que se encuentren errores.)
- Modo **classic**
- 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)
- Los combates serán grabados y devueltos al alumno como feedback por si los quiere revisar.

©19 de octubre de 2025

3.4 Talleres

Taller UD01_T01: 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://iesepm.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:

1. **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.
2. **Análisis estático avanzado:** Proporciona un análisis de código en profundidad que detecta errores y problemas potenciales antes de la compilación.
3. **Depuración avanzada:** ofrece un potente conjunto de herramientas de depuración que ayudan a identificar y resolver problemas en el código.
4. **Refactorización guiada:** Proporciona herramientas para reorganizar y optimizar el código de forma segura, promoviendo buenas prácticas de programación.
5. **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.
6. **Generación automática de código:** ayuda a los programadores a generar automáticamente fragmentos de código repetitivos, como captadores y definidores.
7. **Integración con herramientas de compilación:** facilita la integración con herramientas de compilación como Maven y Gradle.
8. **Sopporte para pruebas unitarias:** Ofrece integración con marcos de prueba como JUnit para el desarrollo basado en pruebas.
9. **Facilidad de configuración:** Proporciona asistentes guiados para configurar de manera eficiente proyectos Java.

Contras:

1. **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.
2. **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 (VS CODE):

Ventajas:

1. **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.
2. **Amplia gama de extensiones:** Tiene una amplia comunidad que desarrolla extensiones para diversas tecnologías y lenguajes, incluido Java.
3. **Versatilidad:** Si bien no está diseñado específicamente para Java, se puede personalizar para que funcione con Java a través de extensiones.
4. **Integración de control de versiones:** ofrece integración nativa con sistemas de control de versiones como Git.
5. **Curva de aprendizaje rápida:** Debido a su enfoque más ligero, puede resultar más sencillo para los principiantes comenzar a trabajar con él.

Contras:

1. **Funcionalidad limitada de Java:** Aunque existen extensiones de Java, VSCode no ofrece el mismo conjunto completo de herramientas optimizadas para Java que IntelliJ IDEA.
2. **Análisis menos profundo:** Las capacidades de análisis estático y corrección de código podrían no ser tan avanzadas como las de IntelliJ IDEA.
3. **Depuración limitada:** si bien ofrece depuración, es posible que no sea tan avanzada o completa como la de IntelliJ IDEA.
4. **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)

⌚17 de septiembre de 2025

Taller UD01_T02: Crear cuenta en GitHub

Qué es GitHub

GitHub es una plataforma en la nube basada en Git que permite a los desarrolladores almacenar, gestionar y colaborar en proyectos de código. Es el portafolio universal de los programadores.

Crear una cuenta es esencial para quien aprende o busca trabajar en programación porque: sirve como tu currículum técnico, donde muestras tus proyectos y evolución; te permite colaborar en proyectos open source para ganar experiencia real; y es una herramienta fundamental para el control de versiones y trabajo en equipo, usada por prácticamente todas las empresas tech.

Crea tu cuenta

Accede a la plataforma GitHub: <https://github.com/>

Pulsa sobre el botón [Sign Up] y sigue las instrucciones para crear tu cuenta.

Una vez creada tu cuenta, entra en tu página principal, por ejemplo la mia es esta: <https://github.com/martinezpenya> (`martinezpenya` es mi usuario de github) y realiza una captura de pantalla.

Solicitar corrección de los apuntes

Ahora, para probar nuestra nueva cuenta y colaborar con algún proyecto, no hay nada mejor que ayudar a mejorar los apuntes del profesor de Programación 😊.

Accedemos a la página de los apuntes en la que hemos detectado el error o queremos sugerir un cambio y en la parte superior derecha debe aparecer el icono:

1º Programacion (CFGs Desarrollo de Aplicaciones Multiplataforma)

UD00

UD01

Elementos de un programa informático

Ejercicios

Talleres

T01 NoMachine

T02 JDK e IDE

T03 GitHub

T04 Markdown

Introducción a Markdown

Para qué sirve Markdown

Dar más información sobre Markdown

Taller UD01_T04: Markdown

1. Introducción a Markdown

M↓

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 v

Esto nos llevará a crear un Fork del repositorio (este concepto lo aprenderás más adelante en el módulo de Entornos de Desarrollo):

martinezpenya / 1DAMProgramacion

Type ⌘ to search

Code Issues Pull requests Actions Projects Security Insights

1DAMProgramacion / docs / UD01 / UD01_T04_Markdown.md in main

You need to fork this repository to propose changes.

Sorry, you're not able to edit this repository directly—you need to fork it and propose your changes from there instead.

Fork this repository

Learn more about forks

Ahora debemos pulsar el botón **[Fork this repository]**, y a continuación veremos el código de la página en nuestro fork que es `MarkDown` (Puedes aprender más sobre `MarkDown` en el Taller 4):

You're making changes in a project you don't have write access to. Submitting a change will write it to a new branch in your fork [profeDAMCarlet/1DAMProgramacion](#), so you can send a pull request.

1DAMProgramacion / docs / UD01 / UD01_T04_Markdown.md in [main](#)

Commit changes...

```

1 # Taller UD01_T04: Markdown
2
3 ## Introducción a Markdown
4
5 
6
7 **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 \[licencia BSD\](https://es.wikipedia.org/wiki/Licencia\_BSD).
8
9 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](http://pandoc.org/) ha extendido enormemente la \[sintaxis original de Markdown\](http://daringfireball.net/projects/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.

```

Ahora debemos buscar el texto a modificar y una vez hayamos cambiado algo del documento se activará el botón [Commit changes...]:

You're making changes in a project you don't have write access to. Submitting a change will write it to a new branch in your fork [profeDAMCarlet/1DAMProgramacion](#), so you can send a pull request.

1DAMProgramacion / docs / UD01 / UD01_T04_Markdown.md in [main](#)

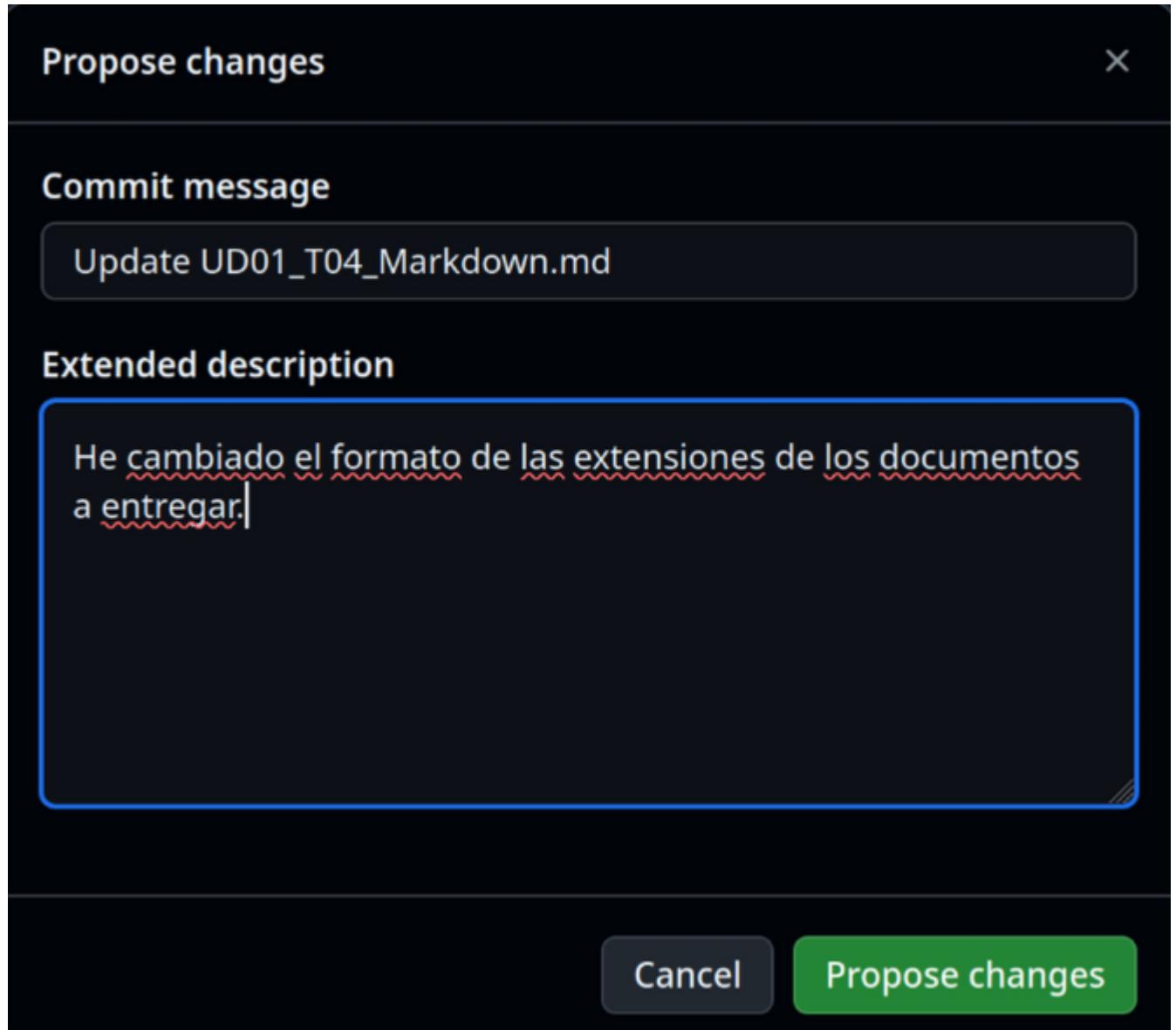
Commit changes...

```

1 # Taller UD01_T04: Markdown
2
3 ## Introducción a Markdown
4
5 
6
7 **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 \[licencia BSD\](https://es.wikipedia.org/wiki/Licencia\_BSD).

```

Ahora debes explicar cual ha sido la modificación que hemos realizado y pulsar el botón [Propose changes]:



Todavía no hemos terminado! ahora hay que comunicar los cambios propuestos en nuestro Fork al propietario del repositorio, para que los visualice y valore si los quiere incluir en la página de documentación. Para ello debemos pulsar el botón [Create pull request]:

The screenshot shows the GitHub interface for comparing changes between two repositories. At the top, it displays the repositories: 'martinezpenya / 1DAMProgramacion' and 'base repository: martinezpenya/1DAMProgramacion...'. Below this, there are dropdown menus for 'base: main' and 'head repository: profeDAMCarlet/1DAMProgramacion...' with 'compare: patch-1'. A message encourages users to 'Discuss and review the changes in this comparison with others.' followed by a link to 'Learn about pull requests'. A prominent green button on the right says 'Create pull request'. Below this, summary statistics are shown: '-o 1 commit', '1 file changed', and '1 contributor'. A detailed commit log shows a single commit from 'profeDAMCarlet' on Sep 6, 2025, titled 'Update UD01_T04_Markdown.md ...'. The commit details include a 'Verified' badge, a copy icon, the hash '5f152ae', and a link. At the bottom, a code diff viewer shows the changes made to 'docs/UD01/UD01_T04_Markdown.md', specifically line 393, column 4, where the text 'Como tarea, se propone:' was modified.

Ahora podemos modificar el mensaje (pero no hace falta), directamente pulsamos sobre el botón [Create pull request]:

The screenshot shows the GitHub interface for creating a pull request. At the top, the repository 'martinezpenya / 1DAMProgramacion' is selected. Below it, the 'Code' tab is active, while other tabs like 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights' are visible. A search bar and various navigation icons are also present.

The main area is titled 'Open a pull request'. It asks to 'Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. [Learn more about diff comparisons here.](#)'

Below this, there are dropdowns for 'base repository' (set to 'martinezpenya/1DAMProgramacion'), 'base' (set to 'main'), and 'head repository' (set to 'profeDAMCarlet/1DAMProgramacion'). The 'compare' dropdown is set to 'patch-1'.

The 'Add a title' field contains the text 'Update UD01_T04_Markdown.md'. To the right, under 'Helpful resources', are links to 'GitHub Community Guidelines' and 'GitHub Help'.

The 'Add a description' section contains the text 'He cambiado el formato de las extensiones de los documentos a entregar.' Below the editor, it says 'Markdown is supported' and 'Paste, drop, or click to add files'.

At the bottom, there is a 'Create pull request' button with a green background and white text. A red arrow points to this button. To its left is a checkbox labeled 'Allow edits by maintainers' with a question mark icon. Below the button, a note says 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)'.

Ahora si, deberías ver una página similar a la siguiente, de la que también deberás obtener una captura y adjuntarla al .pdf , y además explicar los 4 campos que hay redondeados:

Update UD01_T04_Markdown.md #1

profeDAMCarlet wants to merge 1 commit into [martinezpenya:main](#) from [profeDAMCarlet:patch-1](#)

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -1 0 0

profeDAMCarlet commented now

He cambiado el formato de las extensiones de los documentos a entregar.

No conflicts with base branch
Changes can be cleanly merged.

Add a comment

Write Preview

Add your comment here...

Markdown is supported Paste, drop, or click to add files

[Close pull request](#) [Comment](#)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

ProTip! Add `.patch` or `.diff` to the end of URLs for Git's plaintext views.

Reviewers
No reviews
Still in progress? [Convert to draft](#)

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications [Unsubscribe](#)
You're receiving notifications because you authored the thread.

1 participant

Allow edits by maintainers [?](#)

Como resumen:

1. Hemos creado un fork de un repositorio
2. Hemos modificado un archivo en nuestro fork
3. Hemos comparado nuestro fork con el original y hemos creado un pull request con las diferencias

Ahora pueden pasar dos cosas, que el propietario del repositorio original acepte nuestros cambios, y por tanto pasaremos a ser colaboradores del repositorio original.

O bien, que el cambio no sea aceptado.

En cualquiera de los dos casos, si adjuntas las capturas y explicas los campos la actividad estará correcta.

En este caso concreto se ha aceptado la modificación:

Update UD01_T04_Markdown.md #1

Merged martinezpenya merged 1 commit into `martinezpenya:main` from `profeDAMCarlet:patch-1` 1 minute ago

Conversation 1 Commits 1 Checks 0 Files changed 1

profeDAMCarlet commented 10 minutes ago

He cambiado el formato de las extensiones de los documentos a entregar.

martinezpenya commented 1 minute ago

Perfecto, gracias!

martinezpenya closed this 1 minute ago

martinezpenya merged commit `8052475` into `martinezpenya:main` 1 minute ago

Tarea

Crea un documento `.pdf`

Añade una **captura** captura de tu perfil de github.

Añade una **captura** de pantalla donde se vea 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 **4 apartados** señalados en la captura.

Adjunta el documento `.pdf` con las capturas y las explicaciones a la tarea de AULES

7 de octubre de 2025

Taller UD01_T03: Markdown

Introducción a Markdown

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:

1. Encapçalament 1

1.1. Encapçalament 2

1.1.1. Encapçalament 3

1.1.1.1. Encapçalament 4

1.1.1.1.0.1. Encapçalament 5

1.1.1.1.0.1. Encapçalament 6

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 *itálica*.
3 Este texto está ~~tachado~~.
4 Este texto es en ambos ***negrita e itàlica***.
```

Se corresponde a:

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

Este texto es en *itálica*.

Este texto está ~~tachado~~.

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

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 un 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 emojis 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 (Sí o No?)`
- Subrutina: `sub1=>subroutine: nombre subtarea`
- EntradaSalida: `io1=>inputoutput: nombre elemento entrada/salida`
- Líneas: `st->op->cond`
- Caminos de condiciones: `cond(yes)**->**e y cond(no)->op`
- Línea de cierre: `````

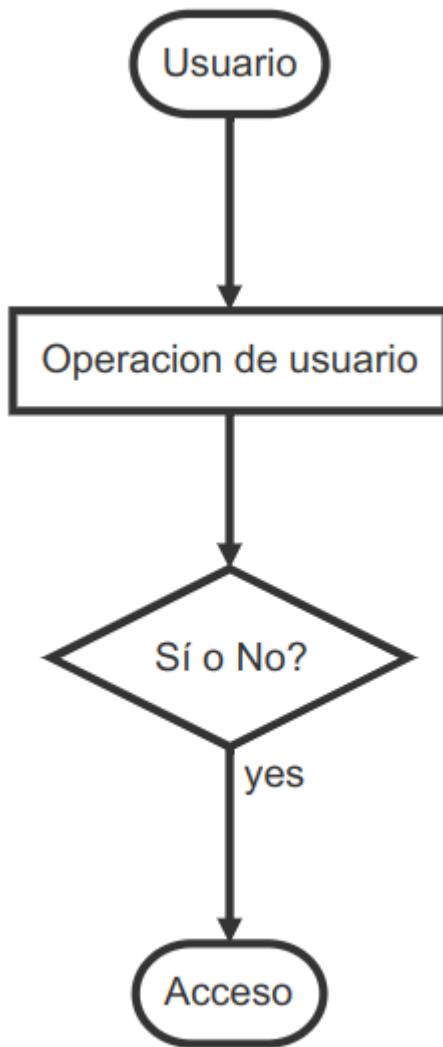
Ejemplo:

```

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


```

Visualización:



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

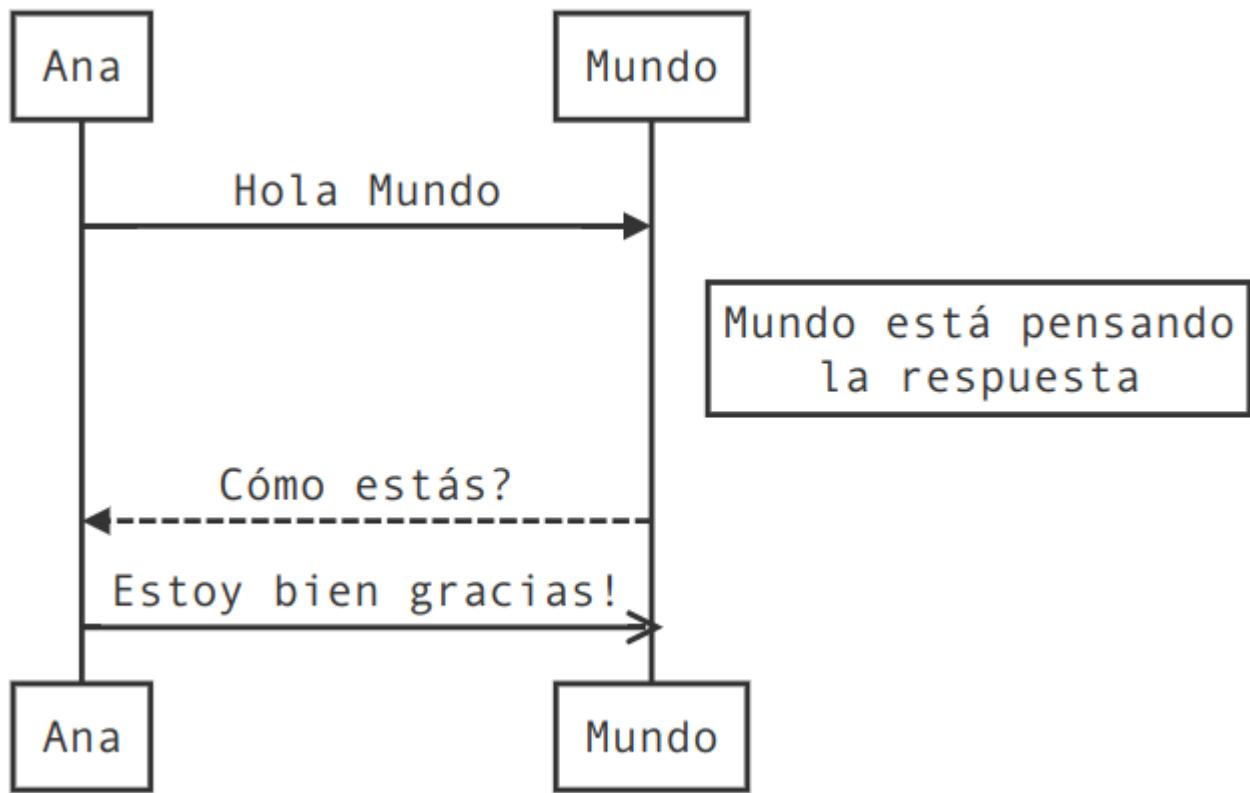
En la secuenciación podemos observar que es bastante parecido a la creación de diagramas; pero la primera línea (crear un bloque de código) no será **flow** sino **sequence**.

```

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:



#### 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.**

⌚17 de septiembre de 2025

## 4. UD02

### 4.1 Sistemas basados en el conocimiento

#### 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 \(\backslash(v\backslash)\) datos \(\backslash(v\backslash)\) información:

- **Datos:** Hechos o valores.

- **Información:** Datos con significado.

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

#### Definición

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  $\rightarrow$  Vertebrados  $\rightarrow$  Mamíferos  $\rightarrow$  Perros  $\rightarrow$  Caniche
- Animales  $\rightarrow$  Vertebrados  $\rightarrow$  Pájaros  $\rightarrow$  Palomas  $\rightarrow$  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:  $\neg p$ : "La persona tiene fiebre",  $\neg q$ : "La persona debe tomar paracetamol"
- $p \rightarrow q$ : "Si la persona tiene fiebre, entonces la persona debe tomar paracetamol"
- $p \wedge q$ : "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».

!!! warning "Importante 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.

```

flowchart TB
 subgraph Motor
 M{Motor de inferencia}
 end
 subgraph Usuario
 U[Usuario] <-- Interfase persona-máquina --> M
 end
 subgraph Experto
 E[Experto]
 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.

## Ejemplo

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:

```

1 IF (cliente mayor de 18 años) AND (tiene nómina)
2 AND ((tiene contrato indefinido) OR (la duración del contrato es superior al tiempo de devolución del préstamo))
3 THEN (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 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.

### Más información

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



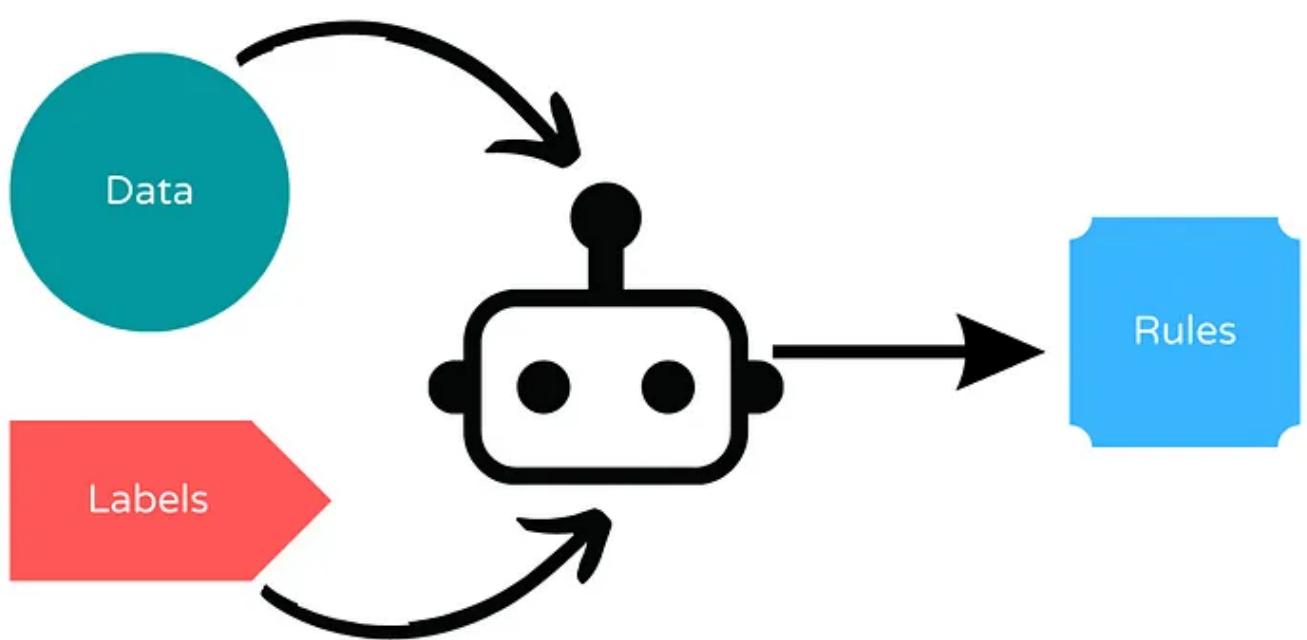
**ENCADENAMIENTO HACIA ADELANTE Y HACIA ATRAS**

Adriana Rodriguez  
Marcela Herrera



### 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:**

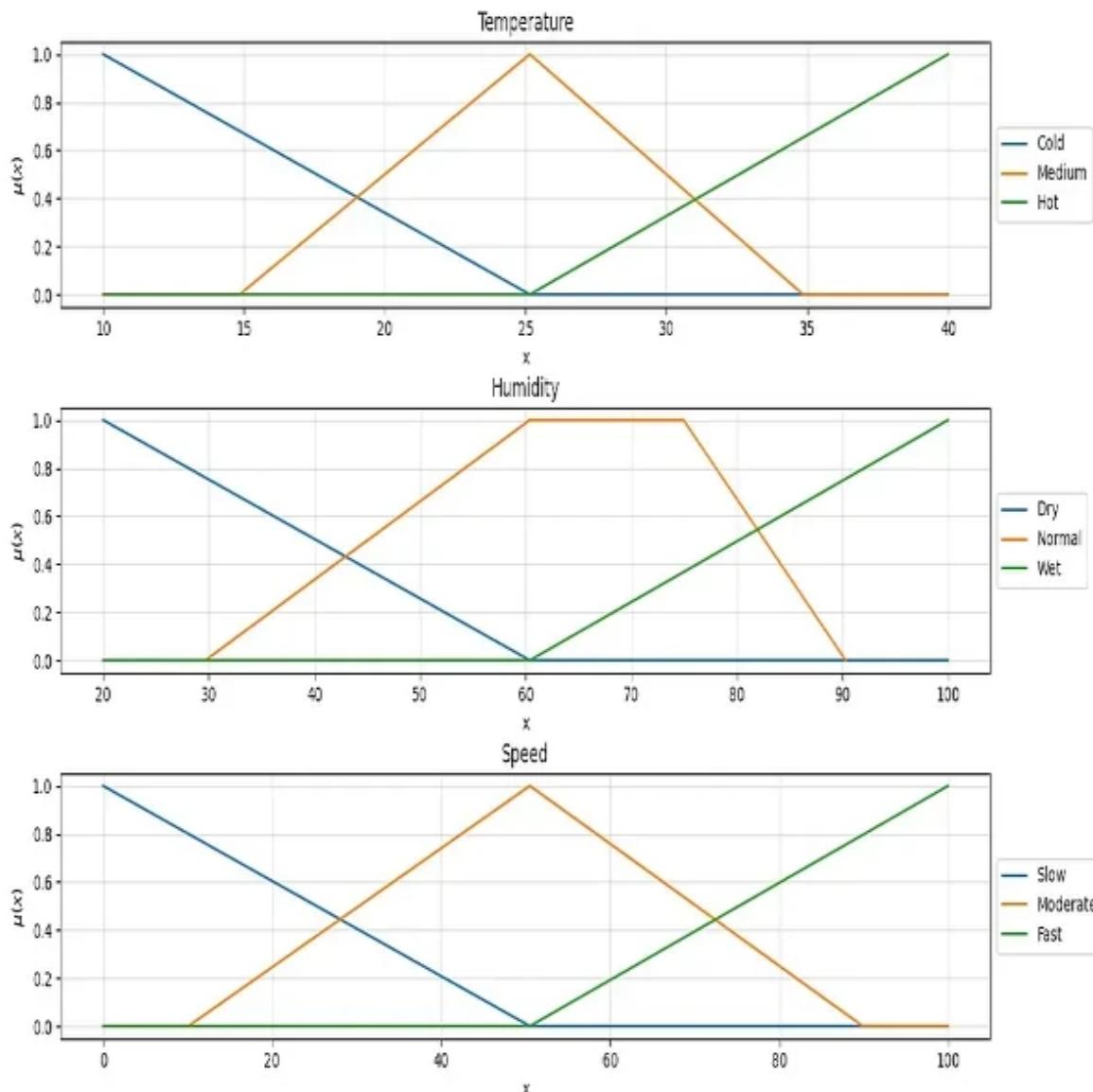
- Extensión de la lógica proposicional para trabajar con la incertidumbre.
- Permite trabajar con valores imprecisos.

**• Sistemas de razonamiento impreciso:**

- Sistemas basados en reglas que utilizan la lógica difusa.
- Permiten trabajar con valores **continuos**.
- Facilitan modelar el **conocimiento humano**.
- Muy apropiados para **sistemas de control**
- Nos permiten tener una **buena** solución, si no la **mejor**.

**Lògica difusa**

- La lógica proposicional es **binaria**.
- Un enunciado es **cierto** o **falso**.
- La lógica 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  $\{[0, 1]\}$ .
- $\{0: \text{Falso}\}, \{1: \text{Cierto}\}, \{0.5: \text{Cert}\}$  en un  $\{50\%\}$
- La pertenencia de un elemento a un conjunto vendrá dada por una **función de pertenencia**.
- $\{\mu_A(x)\}$ : Grado de pertenencia de  $(x)$  al conjunto  $(A)$ .



- 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 \(\text{(húmedo)}\) o \(\text{(frio)}\) 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: \(\text{(Temperatura)}
- **Valores lingüísticos:** Valores que puede tomar una variable lingüística.
- Ejemplo: \(\text{(Frío, Calor)}
- **Función de pertenencia:** Función que asigna a cada valor de una variable lingüística un grado de pertenencia a un valor lingüístico.
- Ejemplo: \(\text{(Temperatura} = 27^{\circ}\text{C} \rightarrow \text{Calor} = 0.8\text{; Combustible} \rightarrow \text{Calor} = 0.2\text{)}
- **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:  $(\text{Calor} = 0.8, \text{Humedo} = 0.7) \rightarrow (\text{Sensación} = 0.8)$
- **Sistema de razonamiento impreciso:**
- Sistema basado en reglas que utiliza la lógica difusa.
- Ejemplo: Sistema de control de temperatura de una casa.

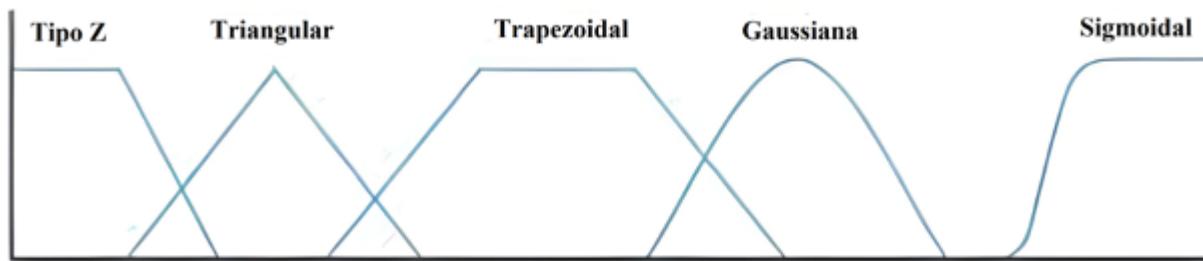
#### Funcionamiento de los sistemas de razonamiento impreciso

- **Fuzzyficación:**
- 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**
- $(27^\circ\text{C} \rightarrow \text{Calor} = 0.8, \text{Mucha} \rightarrow \text{calor} = 0.2)$
- **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**.



- **Desfuzzyficación:**
- 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

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



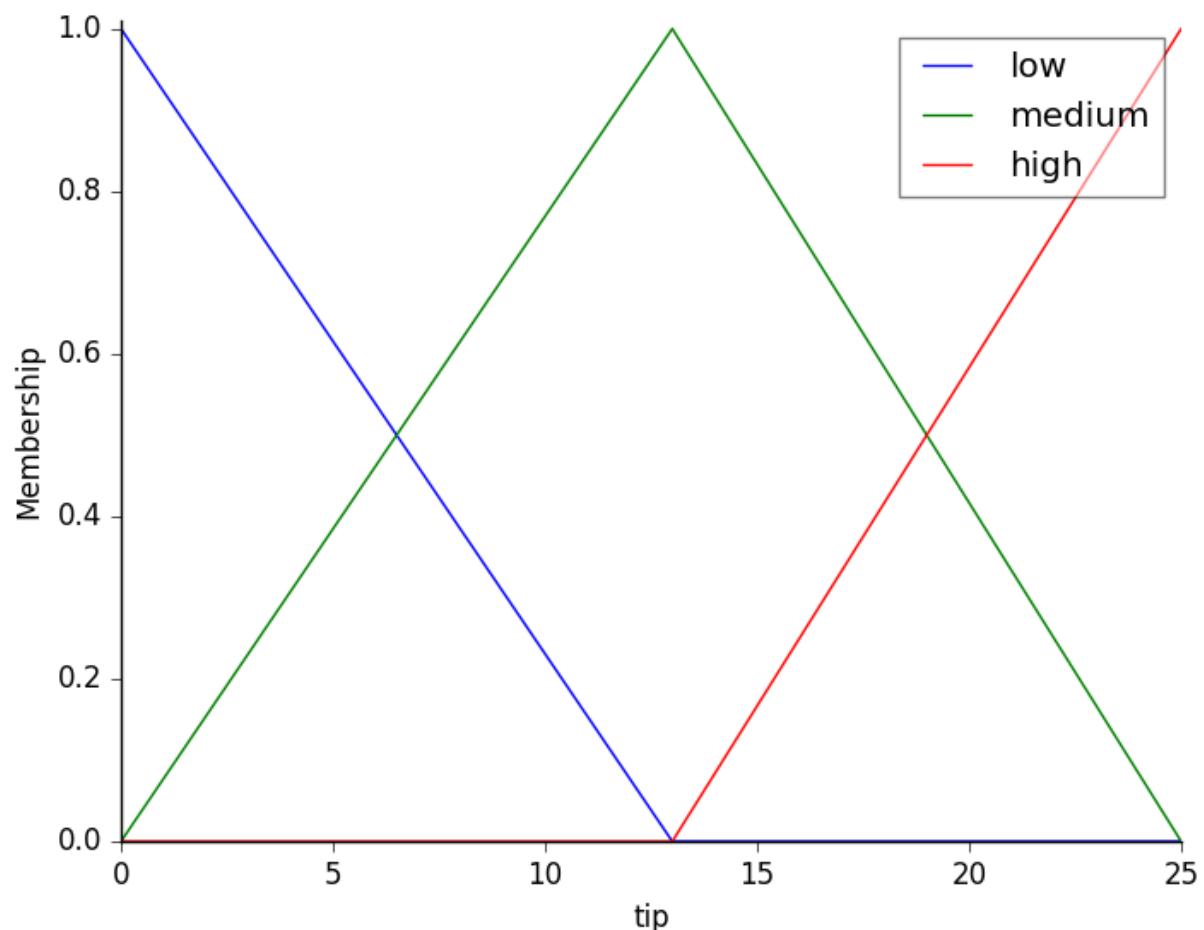
• **Calidad de la comida:**

- **Mala:**  $\lambda([0, 5])$
- **Media:**  $\lambda([0, 10])$
- **Buena:**  $\lambda([5, 10])$

VARIABLES DE SALIDA

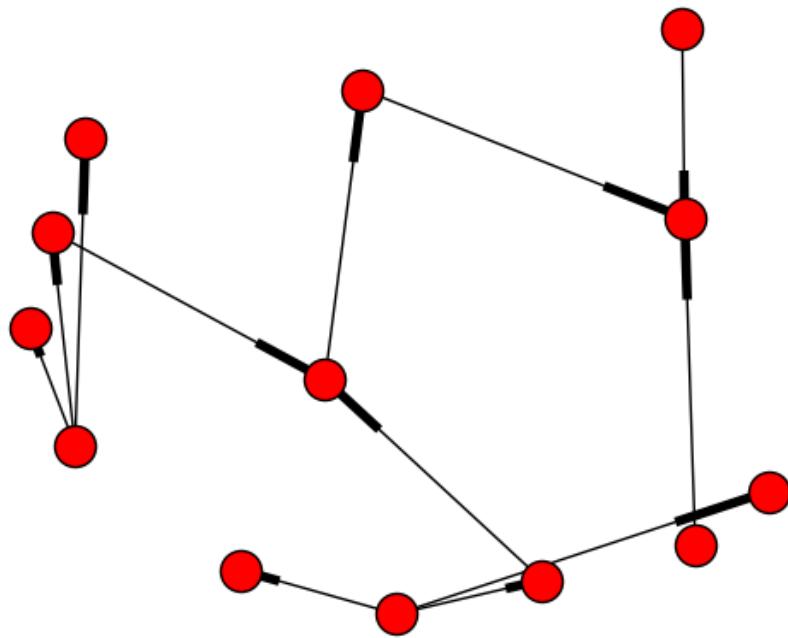
• **Propina:**

- **Baja:**  $\lambda([0, 13])$
- **Media:**  $\lambda([0, 25])$
- **Alta:**  $\lambda([13, 25])$



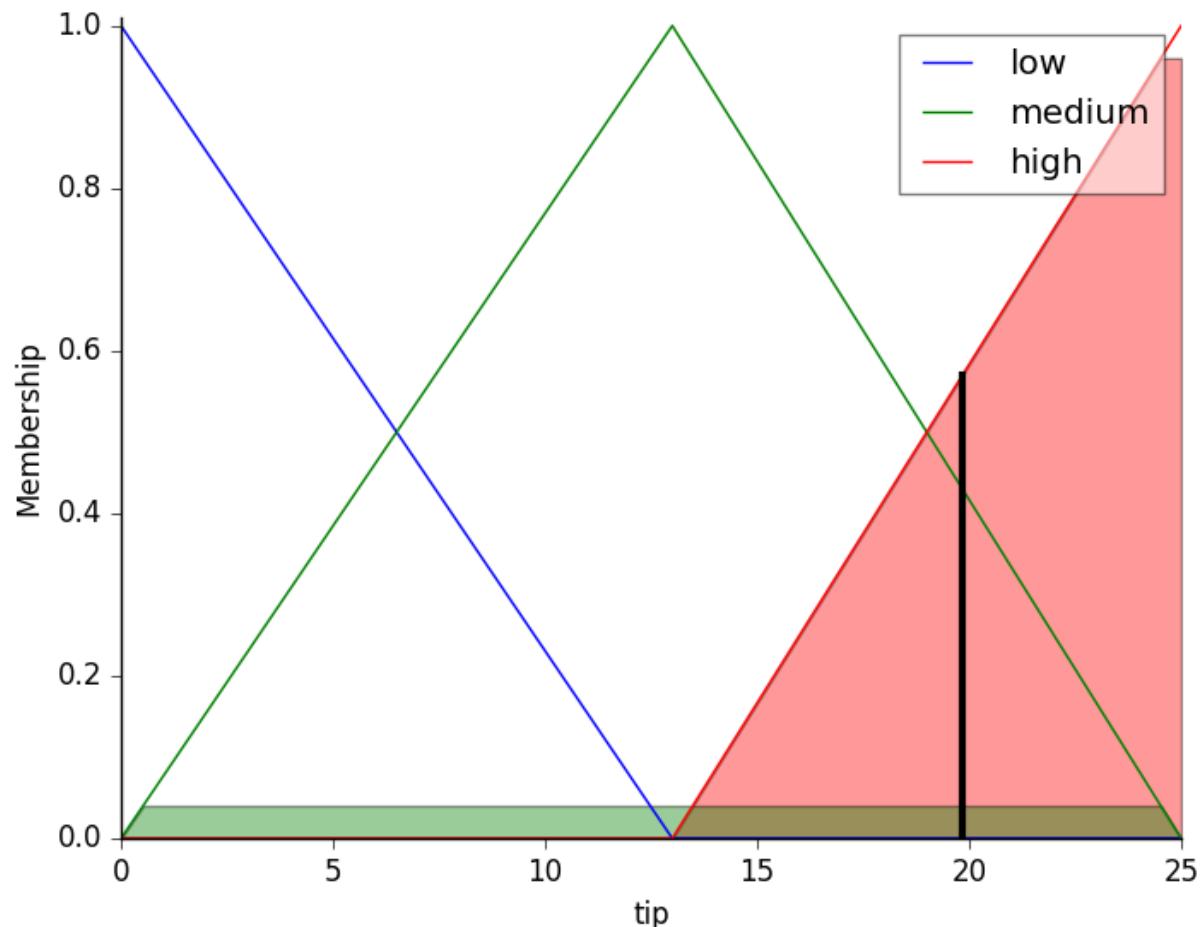
## REGLAS

- **IF** (Calidad del servicio es **baja** o Comida es **mala**) **THEN** (Propina es **baja**)
- **IF** (Calidad del servicio es **media**) **THEN** (Propina es **media**)
- **IF** (Calidad del servicio es **alta** o Comida es **buena**) **THEN** (Propina es **alta**)



## INFERENCIA

- Calidad del servicio: **9.8**
- Calidad de la comida: **6.5**
- Propina: **19,24 €**



#### Más información

**Lógica difusa** Si quieres profundizar más en el tema de la lógica difusa o borrosa que permite la toma de decisiones en algunos sistemas expertos, puedes leer sobre ello en este [artículo de la Wikipedia](#).

23 de noviembre de 2025

## 4.2 Diapositivas de la UD02

---

-  Diapositivas

⌚18 de noviembre de 2025

## 4.3 Actividades guiadas de la UD02

| Notebook                                                                                                                                             | Enlace                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Sistema expertos con Python y Experta                                                                                                                |  Open in Colab |
| Piedra, papel o tijera                                                                                                                               |  Open in Colab |
| Sistema Experto: Clasificación de animales                                                                                                           |  Open in Colab |
| Sistema basado en reglas: Titanic                                                                                                                    |  Open in Colab |
| Sistema de control: Regar el césped                                                                                                                  |  Open in Colab |
| docker-compose.yml y Dockerfile necesarios para ejecutar contenedor con un notebook en local<br>(python 3.10) y resolver el problema de Google Colab | docker-compose.yml<br>Dockerfile                                                                  |
| Lógica difusa: Quien jugará                                                                                                                          |  Open in Colab |

⌚23 de noviembre de 2025

## 4.4 Actividades entregables de la UD02

| Notebooks a entregar:                                                     | Enlace                                                                                                     |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| EX1 Sistema experto: Detectar lesiones de rodilla                         |  Open in Colab          |
| EX2 Sistema basado en reglas: Previsión del valor de mercado de jugadores |  Open in Colab          |
| EX3 Lógica difusa: Detectar centro-campistas jóvenes                      |  Open in Colab          |
| CSV con datos de jugadores del FIFA 22 (Para EX2 y EX3)                   |  CSV EX2 .players 22.csv |
| EX4 Sistema de control: Quemador de gas                                   |  Open in Colab          |
| EX0 Sistema Experto con Experta (enunciado)                               |  Open in Colab          |
| EX0 Ejemplo resuelto por el profesor                                      |  Open in Colab          |

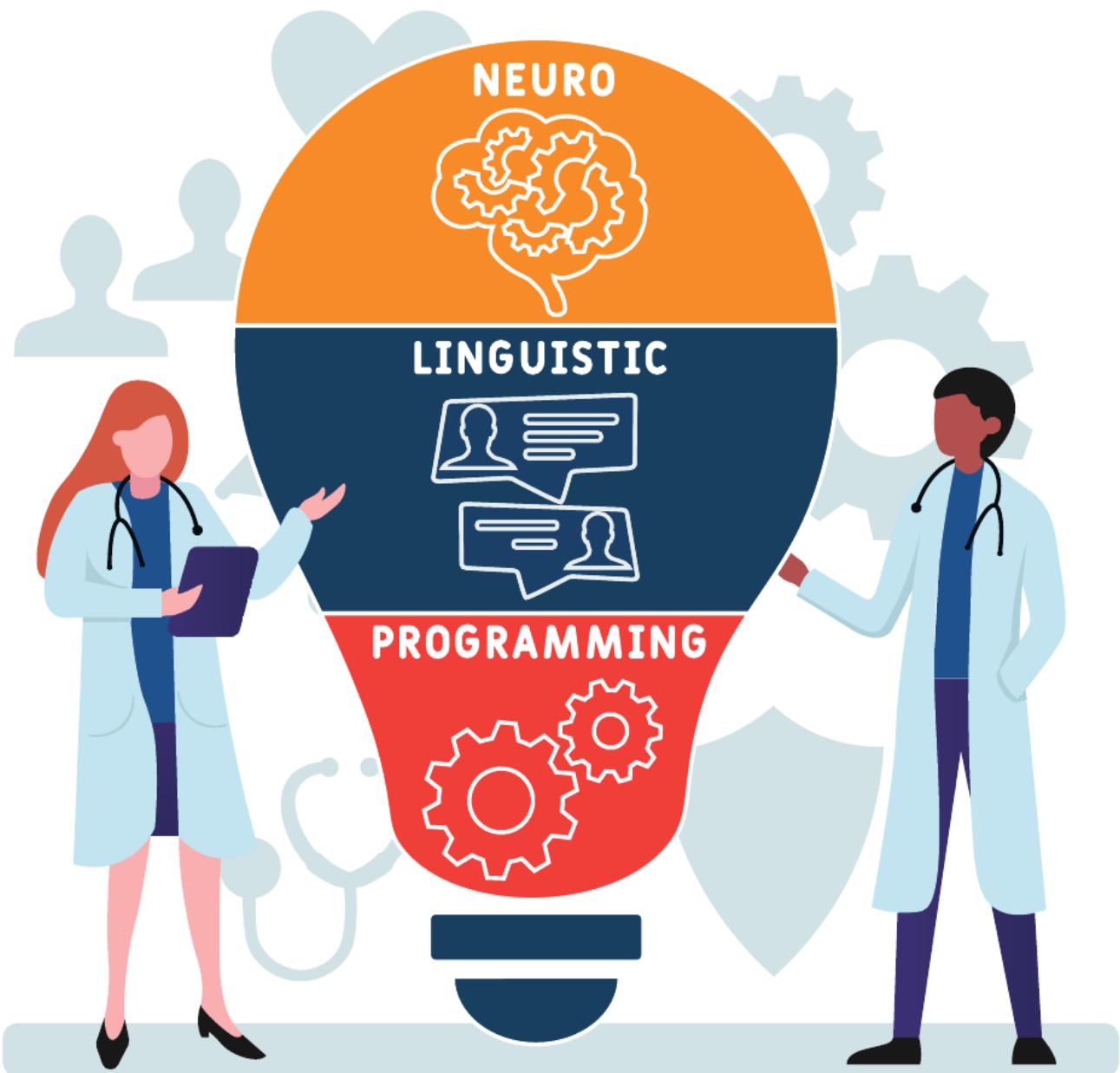
 18 de noviembre de 2025

## 5. UD03

### 5.1 Procesamiento del Lenguaje Natural

#### Introducción al PLN

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

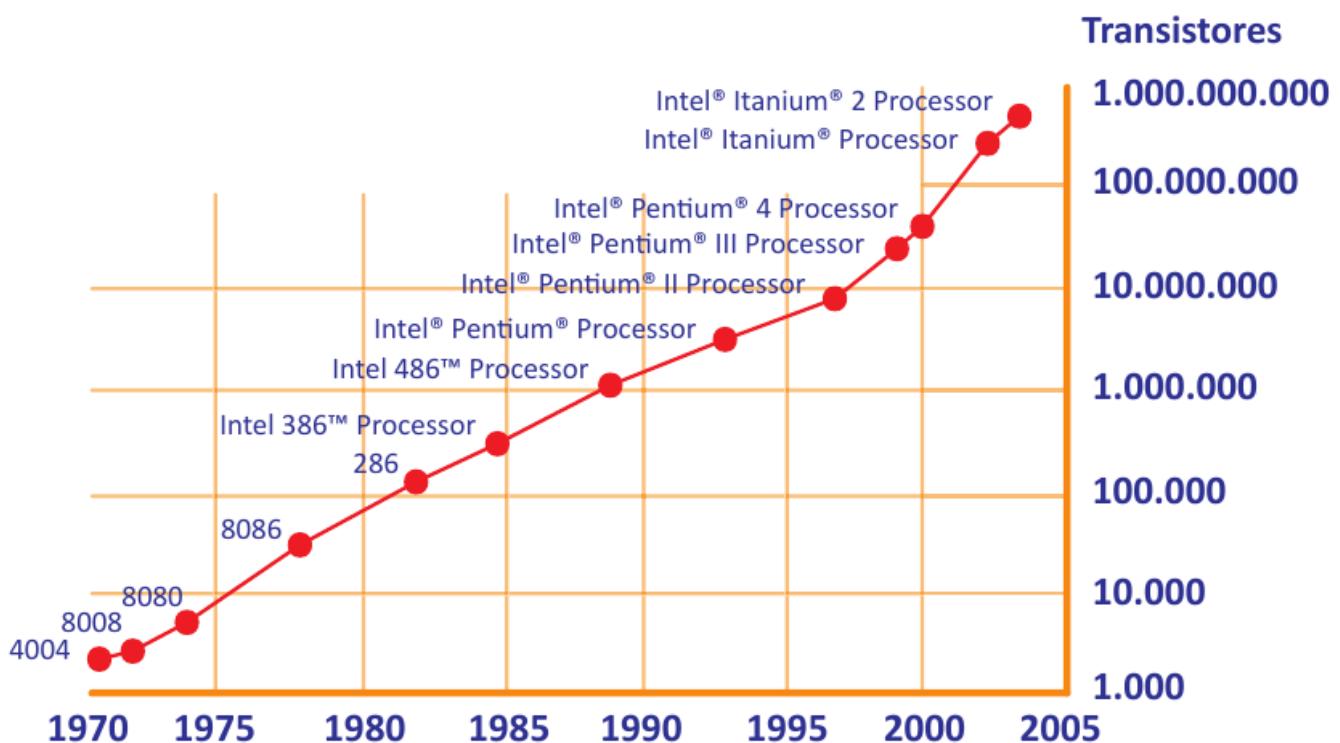


El procesamiento del lenguaje natural estudia las relaciones del lenguaje entre los seres humanos y las máquinas.

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

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

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



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

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

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

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

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

Si una persona española tuviera que pensar qué corpus emplear para el entrenamiento de un procesador de lenguaje natural en español, es fácil que centrara en las grandes obras literarias de nuestra lengua, como por ejemplo *El Quijote*, o un texto difundido en nuestra lengua, de amplia proyección, como puede ser *La Biblia*.

Lo mismo ocurre en otros idiomas, y en inglés en concreto, existe una base de datos que contiene un conjunto de corpus en esta lengua, que fue realizado en 2014 y recibe el nombre de Gutemberg.

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

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

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

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

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

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

- Lingüística Computacional
- Ingeniería Lingüística

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

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

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

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

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

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

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

El siguiente enlace presenta un analizador morfológico de la Biblioteca Virtual Miguel de Cervantes (BVMC) que usa el **corpus Ancora**. <https://data.cervantesvirtual.com/analizador>

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

RESUMEN:

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

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

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

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

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

### Aplicaciones del PLN

Algunas aplicaciones del PLN son:

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

## Técnicas de procesamiento de lenguaje. Limitaciones

---

### Potencial del procesamiento del lenguaje natural

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

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

#### RECONOCIMIENTO DEL HABLA (ASR, AUTOMATIC SPEECH RECOGNITION)

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

#### SÍNTESIS DE TEXTO A VOZ (TTS, TEXT TO SPEECH)

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

#### DETECCIÓN DE ENTIDADES NOMBRADAS (NER, NAMED ENTITY RECOGNITION)

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

#### TRADUCCIÓN AUTOMÁTICA

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

#### SIMILITUD DE TEXTOS

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

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

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

#### ANÁLISIS DEL SENTIMIENTO

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

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

#### Limitaciones. La ambigüedad

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

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

#### Ambigüedades

##### • Ambigüedad sintáctica:

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

##### • Ambigüedad léxica/morfológica:

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

##### • Ambigüedad semántica:

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

##### • Ambigüedad pragmática:

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

##### • Ambigüedad fonológica:

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

- Ambigüedad funcional:

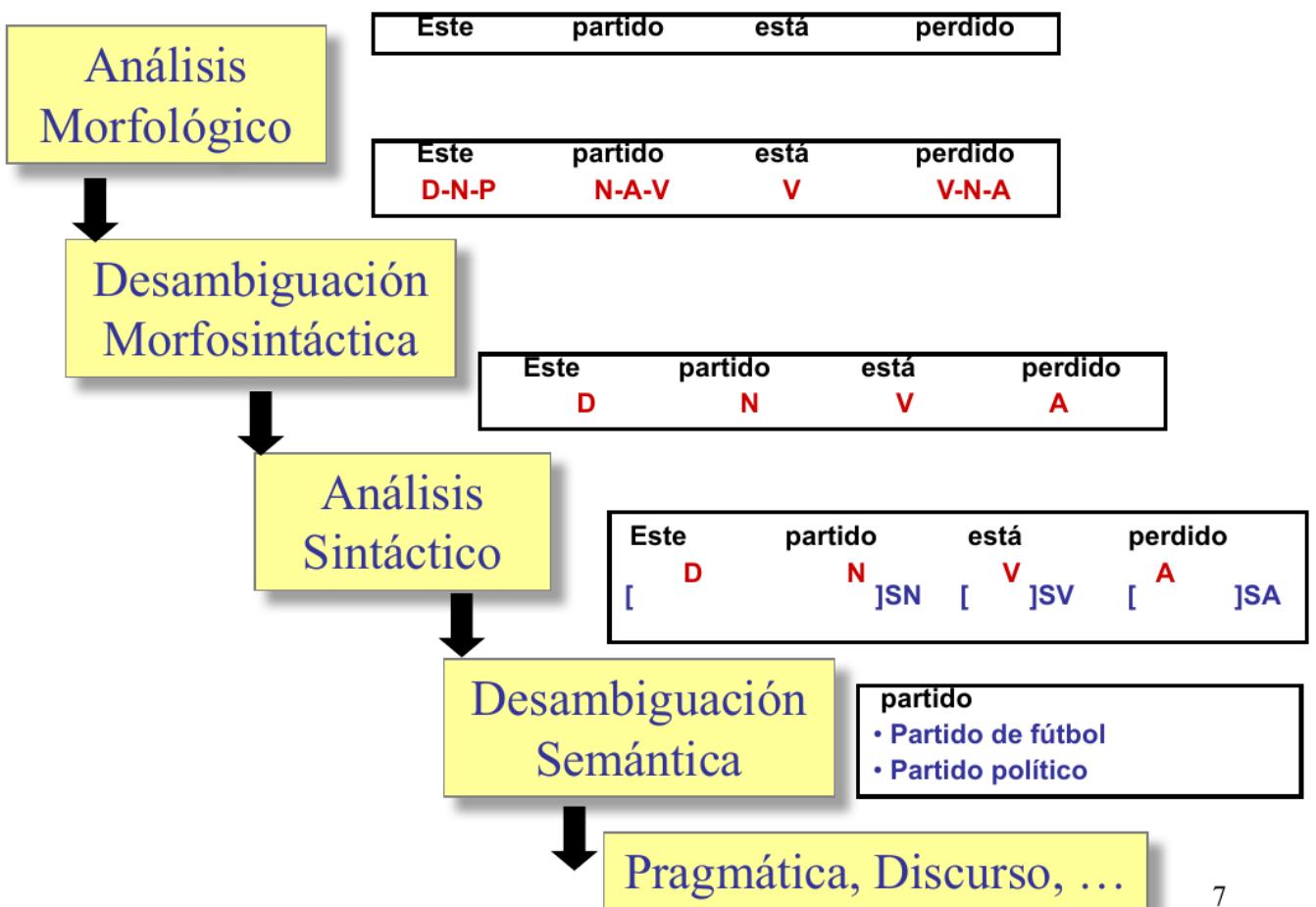
He vuelto a oler (antes no tenía olfato y ahora sí; o bien, regresé a un lugar a oler algo para comprobar su aroma).

#### Desambiguación

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

Ejemplos:

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



#### POS Tagging

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

### Ejemplo

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

### Categorías Gramaticales (POS)

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

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

#### Tipos categorías

- Abiertas o Cerradas

#### Principales Categorías

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

#### Ejemplos de Etiquetas (POS)

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

#### Conjunto de etiquetas **Penn Tree Bank**

| Tag  | Description           | Example                | Tag  | Description           | Example            |
|------|-----------------------|------------------------|------|-----------------------|--------------------|
| CC   | Coordin. Conjunction  | <i>and, but, or</i>    | SYM  | Symbol                | +,%,&              |
| CD   | Cardinal number       | <i>one, two, three</i> | TO   | "to"                  | <i>to</i>          |
| DT   | Determiner            | <i>a, the</i>          | UH   | Interjection          | <i>ah, oops</i>    |
| EX   | Existential 'there'   | <i>there</i>           | VB   | Verb, base form       | <i>eat</i>         |
| FW   | Foreign word          | <i>mea culpa</i>       | VBD  | Verb, past tense      | <i>ate</i>         |
| IN   | Preposition/sub-conj  | <i>of, in, by</i>      | VBG  | Verb, gerund          | <i>eating</i>      |
| JJ   | Adjective             | <i>yellow</i>          | VBN  | Verb, past participle | <i>eaten</i>       |
| JJR  | Adj., comparative     | <i>bigger</i>          | VBP  | Verb, non-3sg pres    | <i>eat</i>         |
| JJS  | Adj., superlative     | <i>wildest</i>         | VBZ  | Verb, 3sg pres        | <i>eats</i>        |
| LS   | List item marker      | <i>1, 2, One</i>       | WDT  | Wh-determiner         | <i>which, that</i> |
| MD   | Modal                 | <i>can, should</i>     | WP   | Wh-pronoun            | <i>what, who</i>   |
| NN   | Noun, sing. or mass   | <i>llama</i>           | WP\$ | Possessive wh-        | <i>whose</i>       |
| NNS  | Noun, plural          | <i>llamas</i>          | WRB  | Wh-adverb             | <i>how, where</i>  |
| NNP  | Proper noun, singular | <i>IBM</i>             | \$   | Dollar sign           | \$                 |
| NNPS | Proper noun, plural   | <i>Carolinas</i>       | #    | Pound sign            | #                  |
| PDT  | Predeterminer         | <i>all, both</i>       | "    | Left quote            | (‘ or “)           |
| POS  | Possessive ending     | 's                     | "    | Right quote           | (’ or ”)           |
| PP   | Personal pronoun      | <i>I, you, he</i>      | (    | Left parenthesis      | ( [, (, {, <)      |
| PP\$ | Possessive pronoun    | <i>your, one's</i>     | )    | Right parenthesis     | ( ], ), }, >)      |
| RB   | Adverb                | <i>quickly, never</i>  | ,    | Comma                 | ,                  |
| RBR  | Adverb, comparative   | <i>faster</i>          | .    | Sentence-final punc   | (. ! ?)            |
| RBS  | Adverb, superlative   | <i>fastest</i>         | :    | Mid-sentence punc     | (: ; ... --)       |
| RP   | Particle              | <i>up, off</i>         |      |                       |                    |

Conjunto reducido de etiquetas **Parole**

|                                         |                                                      |
|-----------------------------------------|------------------------------------------------------|
| <b>AQ</b> Adjetivos                     | <b>VAC</b>                                           |
| <b>C0</b> Conjunción sin clasificar     | <b>V</b> Verbo A Auxiliar                            |
| <b>CC</b> Conjunción Coordinada         | <b>C</b> Condicional                                 |
| <b>CS</b> Conjunción Subordinada        | <b>VAG</b> G Gerundio                                |
| <b>D0</b> Determinante sin clasificar   | <b>VAI</b> I Otros tiempos de indicativo             |
| <b>DD</b> Determinante Demostrativo     | <b>VAM</b> M Imperativo                              |
| <b>DE</b> Determinante Exclamativo      | <b>VAN</b> N Infinitivo                              |
| <b>DI</b> Determinantes Indefinidos     | <b>VAP</b> P Particípio                              |
| <b>DP</b> Determinante Posesivo         | <b>VAS</b> S Subjuntivo                              |
| <b>DT</b> Determinante Interrogativo    | <b>VMC</b> M Principal                               |
| <b>E0</b> Términos Extranjeros          | <b>VMG</b>                                           |
| <b>I</b> Interjecciones                 | <b>VMI</b>                                           |
| <b>MC</b> Numeral Cardinal              | <b>VMM</b>                                           |
| <b>MO</b> Numeral Ordinal               | <b>VMN</b>                                           |
| <b>NC</b> Nombre Común                  | <b>VMP</b>                                           |
| <b>NP</b> Nombre Propio                 | <b>VMS</b>                                           |
| <b>P0</b> Pronombre sin clasificar      | <b>W</b> Fecha                                       |
| <b>PD</b> Pronombre Demostrativo        | <b>X</b> Residuales                                  |
| <b>PI</b> Pronombre Indefinido          | <b>Y</b> Abreviaturas                                |
| <b>PP</b> Pronombre personal            | <b>Z</b> Cifras                                      |
| <b>PR</b> Pronombre Relativo            | <b>SIGNOS DE PUNTUACIÓN</b>                          |
| <b>PT</b> Pronombre Interrogativo       | <b>Faa</b> ; <b>Fah</b> [ <b>Fai</b> ;               |
| <b>PX</b> Pronombre Posesivo            | <b>Fal</b> { <b>Fap</b> ( <b>Fc</b> ,                |
| <b>RG</b> Adverbios y Fases Adverbiales | <b>Fca</b> ! <b>Fcd</b> " <b>Fch</b> ]               |
| <b>SP</b> Preposiciones                 | <b>Fci</b> ? <b>Fcl</b> } <b>Fcp</b> )               |
| <b>TD</b> Artículos                     | <b>Fcs</b> ' <b>Fdp</b> : <b>Fg</b> -                |
| <b>TI</b> Determinante Indefinido       | <b>Fgd</b> – <b>Fp</b> . <b>Fpc</b> ; <b>Fps</b> ... |
|                                         | <b>Fs</b> / <b>Ftp</b> % <b>Fac</b> << <b>Fcc</b> >> |

## Formación del investigador en PLN

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

- **NLTK:** puede ser descargado desde el portal web propietario <https://www.nltk.org/> y a partir de ahí leer las instrucciones de instalación <https://www.nltk.org/install.html>, que son muy simples, y descargar los sets de datos mediante el comando `nltk.download(all)` dentro del entorno de Python 3 (y tras ejecutar el import correspondiente). Incluso puede instalar el Stanford CoreNLP API para NLTK desde <https://github.com/nltk/nltk/wiki/Stanford-CoreNLP-API-in-NLTK> que incluye modelo para nuestra lengua. Dentro de NLTK debería seguir el orden lógico que se muestra en el NLTK-BOOK público (disponible en <https://www.nltk.org/book/>):
- Cargar un corpus.
- Usar expresiones regulares.
- Tokenizar y etiquetar.
- Emplear lo anterior para hacer etiquetado POS.
- Usar diccionarios de eliminación de stop-words.

- Usar N-gramas.
- Lematizar.
- Programar un clasificador (por ejemplo, de películas según temática o de noticias).
- Programar un comparador de textos.
- Programar un analizador de sentimientos.
- Programar un sistema de ideas clave o resumen.
- Emplear un TTS (no desde NLTK).
- Emplear un ASR (no desde NLTK).
- Programar un procesador semántico real.
- **Spacy:** puede ser descargado e instalado siguiendo las instrucciones de <https://spacy.io/>. En él se debería practicar lo anterior, además de:
  - Vectorización y comparativa por vectorización a partir de modelos pre entrenados disponibles en el programa como los empleados en este texto.
  - Lo mismo con otros modelos entrenados, como variaciones de BERT.
  - Generar un modelo propio de vectorizado por etiquetación, con un set muy cerrado, de tal manera que las similitudes sean más específicas al texto que se esté tratando.
  - NER, y análisis morfológico con visualización de grafos a través de displacy.
  - Traducción entre idiomas.
- **NeMo:** ha de ser empleado de manera global para NLP, TTS y ASR siguiendo los tutoriales y ejemplos de su GitHub, destacándose entre otros, los siguientes:
  - ASR:
    - Detección de voz (Voice activity detection) a través de micrófono
    - Voz a Texto en PC x86, offline.
    - Voz a Texto en PC x86, online.
    - Voz a texto en Jetbot.
  - Comandos de voz (la base para un asistente).
  - TTS.
  - NLP:
    - Lo mismo que en Spacy.
    - Uso del modelo de Megatron.
    - Programar un Q&A tipo Jeopardy.
- **jetson-voice:** es una biblioteca de inferencia de aprendizaje profundo ASR/NLP/TTS para Jetson Nano, TX1/TX2, Xavier NX y AGX Xavier. Es compatible con Python y JetPack 4.4.1 o posterior. Los modelos DNN se entrenaron con NeMo y se implementaron con TensorRT para optimizar el rendimiento. Todos los cálculos se realizan utilizando la GPU integrada. Actualmente se incluyen las siguientes capacidades:
  - Reconocimiento automático de voz (ASR)
  - Transmisión ASR (QuartzNet)
  - Reconocimiento de comandos/palabras clave (MatchboxNet)
  - Detección de actividad de voz (VAD Marblenet)
  - Procesamiento del lenguaje natural (PNL)
  - Clasificación de intención conjunta/espacio
  - Clasificación de texto (análisis de sentimiento)
  - Clasificación de tokens (reconocimiento de entidad nombrada)
  - Preguntas/Respuestas (QA)
  - Texto a voz (TTS)

## Elaboración de un sistema de procesamiento de lenguaje orientado a una tarea específica

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

- El robot puede cocinar mediante calor (cocer). Esta operación requiere de una temperatura en grados Celsius y de un tiempo expresado en minutos, hasta un máximo de 120 °C.

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

El abordaje de esta labor es sencillo:

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

Puede emplear los contenedores de nVidia para, mediante Docker, emplear NeMo sobre un sistema operativo Ubuntu con gran facilidad.

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

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

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

⌚15 de julio de 2025

## 5.2 Diapositivas de la UD03

---

-  Diapositivas

⌚ 15 de julio de 2025

## 5.3 Actividades guiadas de la UD03

| Notebook                                | Enlace                                                                                          |
|-----------------------------------------|-------------------------------------------------------------------------------------------------|
| Introducción a PLN                      |  Open in Colab |
| Clasificación de texto con PyTorch      |  Open in Colab |
| Analisis de sentimientos con DistilBERT |  Open in Colab |
| Analisis de sentimientos con SpaCy      |  Open in Colab |
| Clasificador de generos musicales       |  Open in Colab |

⌚15 de julio de 2025

## 5.4 Actividades entregables de la UD03

| Notebooks a entregar:                                    | Enlace                                                                                                |
|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| EX1 Representación texto                                 |  Open in Colab       |
| EX2 Clasificador de preguntas                            |  Open in Colab       |
| EX2.5 NLTK y Python                                      |  Open in Colab       |
| EX3 Analisis de Sentimientos IMDB                        |  Open in Colab       |
| EX4 Asistente virtual                                    |  Open in Colab       |
| WAV con el comando de ejemplo "Open the door" (Para EX4) |  WAV OpenTheDoor.wav |

⌚15 de julio de 2025

## 6. UD04

---

### 6.1 Análisis de sistemas robotizados

---

#### Robots

Los robots son agentes físicos que realizan tareas manipulando el mundo físico. Para ello, están equipados con efectores como patas, ruedas, articulaciones y pinzas. Los efectores están diseñados para ejercer fuerzas físicas sobre el medio ambiente. Cuando hacen esto, pueden suceder algunas cosas: el estado del robot puede cambiar (por ejemplo, un automóvil hace girar sus ruedas y, como resultado, avanza en la carretera), el entorno puede cambiar (por ejemplo, un brazo robótico usa su agarre para empujar una taza a través del mostrador), e incluso el estado de las personas alrededor del robot podría cambiar (por ejemplo, un exoesqueleto se mueve y eso cambia la configuración de la pierna de una persona; o un robot móvil avanza hacia las puertas del ascensor, y una persona se da cuenta y es lo suficientemente amable como para apartarse del camino o incluso presionar el botón del robot).

Los robots también están equipados con sensores que les permiten percibir su entorno. La robótica actual emplea un conjunto diverso de sensores, que incluyen cámaras, radares, láseres y micrófonos para medir el estado del medio ambiente y de las personas que lo rodean; y giroscopios, sensores de tensión y torsión, y acelerómetros para medir el propio estado del robot.

Maximizar la utilidad esperada de un robot significa elegir cómo activar sus efectores para hacer valer las fuerzas físicas correctas, aquellas que conducirán a cambios de estado que acumularán la mayor recompensa esperada posible. En última instancia, los robots intentan realizar alguna tarea en el mundo físico.

Los robots operan en entornos que son parcialmente observables y [estocásticos](#): las cámaras no pueden ver en las esquinas y los engranajes pueden patinar. Además, las personas que actúan en ese mismo entorno son impredecibles, por lo que el robot necesita hacer predicciones sobre ellas.

Los robots suelen modelar su entorno con un espacio de estado continuo (la posición del robot tiene coordenadas continuas) y un espacio de acción continuo (la cantidad de corriente que un robot envía a su motor también se mide en unidades continuas). Algunos robots operan en espacios de grandes dimensiones: los coches necesitan conocer la posición, orientación y velocidad de ellos mismos y de los agentes cercanos; los brazos del robot tienen seis o siete articulaciones y cada una se puede mover de forma independiente; y los robots que imitan el cuerpo humano tienen cientos de articulaciones.

El aprendizaje robótica está limitado porque el mundo real se niega obstinadamente a operar más rápido que el tiempo real. En un entorno simulado, es posible utilizar algoritmos de aprendizaje (como el [algoritmo Q-learning](#)) para aprender en unas pocas horas a partir de millones de pruebas. En un entorno real, llevar a cabo estas pruebas podría llevar años y el robot no puede arriesgarse (y por lo tanto no puede aprender) a una prueba que podría causar daño. Por lo tanto, transferir lo aprendido en la simulación a un robot real en el mundo real (el problema de simulación a real) es un área activa de investigación. Los sistemas robóticos prácticos deben incorporar conocimientos previos sobre el robot, el entorno físico y las tareas a realizar para que el robot pueda aprender rápidamente y desempeñarse de forma segura.

La robótica reúne muchos de los conceptos, incluida la estimación probabilística del estado, la percepción, la planificación, el aprendizaje no supervisado, el aprendizaje por refuerzo y la teoría de juegos. Para algunos de estos conceptos, la robótica sirve como un ejemplo de aplicación desafiante.

#### Hardware de Robots

Hasta ahora nos hemos concentrado en el programa del agente. Pero el éxito de los robots reales depende al menos en la misma medida del diseño de sensores y efectores que sean apropiados para la tarea.

##### Tipos de robots desde la perspectiva del hardware

Cuando piensas en un robot, puedes imaginar algo con una cabeza y dos brazos, moviéndose sobre piernas o ruedas. Estos robots [antropomórficos](#) se han popularizado en ficción como la película [The Terminator](#) y la caricatura [Los Supersónicos](#). Pero los robots reales tienen muchas formas y tamaños.

Los [manipuladores](#) son sólo brazos robóticos. No necesariamente tienen que estar unidos al cuerpo de un robot; podrían simplemente atornillarse a una mesa o al suelo, como ocurre en las fábricas.



Algunos tienen una gran carga útil, como los que ensamblan automóviles, mientras que otros, como los brazos montables en sillas de ruedas que ayudan a las personas con discapacidades motoras, pueden transportar menos pero son más seguros en entornos humanos.

Los robots móviles son aquellos que utilizan ruedas, patas o rotores para moverse por el entorno. Los drones cuadricóptero son un tipo de vehículo aéreo no tripulado ([UAV](#)); Los vehículos submarinos autónomos ([AUV](#)) recorren los océanos. Pero muchos robots móviles permanecen en el interior y se mueven sobre ruedas, como una aspiradora o un robot repartidor de toallas en un hotel. Sus homólogos al aire libre incluyen coches autónomos o rovers que exploran nuevos terrenos, incluso en la superficie de Marte.

Nieuwe Marsrover van NASA stuurt adembenemende eerste foto's

Finalmente, los robots con patas están destinados a atravesar terrenos accidentados a los que no se puede acceder con ruedas. La desventaja es que controlar las piernas para hacer lo correcto es más desafiante que hacer girar las ruedas.

Otros tipos de robots incluyen prótesis, exoesqueletos, robots con alas, enjambres y entornos inteligentes en los que el robot es toda la habitación.

### Sintiendo el mundo

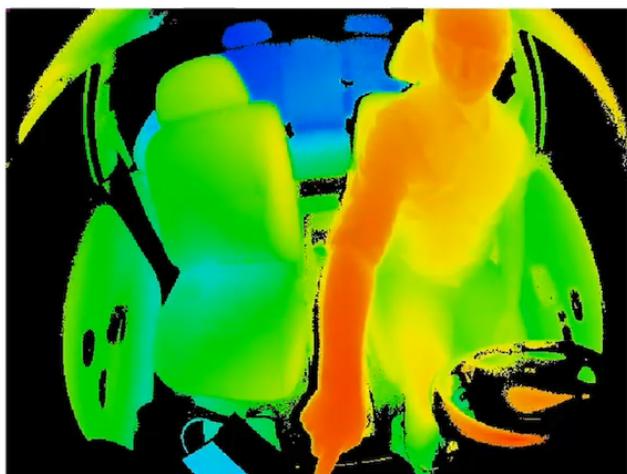
Los sensores son la interfaz perceptiva entre el robot y el entorno. Los **sensores pasivos**, como las cámaras, son verdaderos observadores del entorno: capturan señales generadas por otras fuentes del entorno. Los **sensores activos**, como el sonar, envían energía al medio ambiente. Se basan en el hecho de que esta energía se refleja de vuelta al sensor. Los sensores activos tienden a proporcionar más información que los sensores pasivos, pero a costa de un mayor consumo de energía y con el peligro de interferencias cuando se utilizan varios sensores activos al mismo tiempo. También distinguimos si un sensor está dirigido a **detectar el entorno, la ubicación del robot o la configuración interna del robot**.

Los telémetros son sensores que miden la distancia a objetos cercanos. Los sensores de sonar son telémetros activos que emiten ondas sonoras direccionales, que son reflejadas por los objetos y parte del sonido regresa al sensor. El tiempo y la intensidad de la señal de retorno indican la distancia a los objetos cercanos. El sonar es la tecnología elegida para los vehículos submarinos autónomos y fue popular en los primeros días de la robótica de interior. La [visión estereoscópica](#) se basa en múltiples cámaras para obtener imágenes del entorno desde puntos de vista ligeramente diferentes, analizando el paralaje resultante en estas imágenes para calcular el rango de los objetos circundantes.

Para los robots terrestres móviles, el sonar y la visión estéreo rara vez se utilizan ahora, porque no son confiablemente precisos. El Kinect es un sensor popular de bajo costo que combina una cámara y un proyector de luz estructurada, que proyecta un patrón de líneas de cuadrícula en una escena. La cámara ve cómo se doblan las líneas de la cuadrícula, dándole al robot información sobre la forma de los objetos en la escena. Si se desea, la proyección puede ser luz infrarroja, para no interferir con otros sensores (como los ojos humanos).

[Kinect for Windows SDK Programming Guide | Packt](#)

La mayoría de los robots terrestres ahora están equipados con telémetros ópticos activos. Al igual que los sensores de sonar, los sensores de alcance óptico emiten señales activas (luz) y miden el tiempo hasta que un reflejo de esta señal llega al sensor. La [cámara de tiempo de vuelo](#) adquiere imágenes de rango como la que se muestra más abajo a hasta 60 fotogramas por segundo. Los automóviles autónomos suelen utilizar [lidares](#) de escaneo (abreviatura de detección de luz y alcance): sensores activos que emiten rayos láser y detectan el haz reflejado, brindando mediciones de alcance con una precisión de un centímetro a una distancia de 100 metros. Utilizan complejas disposiciones de espejos o elementos giratorios para barrer el haz a través del entorno y construir un mapa. Los lidars de escaneo tienden a funcionar mejor que las cámaras de tiempo de vuelo a distancias más largas y tienden a funcionar mejor a plena luz del día.



ToF 3D depth image: unaffected by sunlight  
color = distance



ToF 2D amplitude image: unaffected by sunlight

El radar suele ser el sensor de determinación de distancia elegido para los vehículos aéreos (autónomos o no). Los sensores de radar pueden medir distancias de hasta kilómetros y tienen la ventaja sobre los sensores ópticos de que pueden ver a través de la niebla. En el extremo cercano del rango de detección se encuentran los sensores táctiles, como bigotes, paneles de protuberancia y piel sensible al tacto. Estos sensores miden el alcance en función del contacto físico y solo pueden implementarse para detectar objetos muy cerca del robot.

Una segunda clase importante son los sensores de ubicación. La mayoría de los sensores de ubicación utilizan la detección de alcance como componente principal para determinar la ubicación. En exteriores, el Sistema de Posicionamiento Global (GPS) es la solución más común al problema de localización. El GPS mide la distancia a los satélites que emiten señales pulsadas. Actualmente hay 31 satélites [GPS](#) operativos en órbita y 24 satélites [GLONASS](#), su homólogo ruso. Los receptores GPS pueden recuperar la distancia a un satélite analizando cambios de fase. Triangulando señales de múltiples satélites, los receptores GPS pueden determinar su ubicación absoluta en la Tierra con una precisión de unos pocos metros. El GPS diferencial implica un segundo receptor terrestre con ubicación conocida, que proporciona precisión milimétrica en condiciones ideales.

Desafortunadamente, el GPS no funciona en interiores ni bajo el agua. En interiores, la localización a menudo se logra colocando balizas en el entorno en lugares conocidos. Muchos entornos interiores están llenos de estaciones base inalámbricas, que pueden ayudar a los robots a localizar mediante el análisis de la señal inalámbrica. Las balizas de sonar activas bajo el agua pueden proporcionar una sensación de ubicación, utilizando el sonido para informar a los AUV de sus distancias relativas a esas balizas.

La tercera clase importante son los sensores propioceptivos, que informan al robot de su propio movimiento. Para medir la configuración exacta de una articulación robótica, los motores suelen estar equipados con decodificadores de eje que miden con precisión el movimiento angular de un eje. En los brazos de los robots, los decodificadores de ejes ayudan a rastrear la posición de las articulaciones. En los robots móviles, los decodificadores de eje informan las revoluciones de las ruedas para la odometría (la medición de la distancia recorrida). Desafortunadamente, las ruedas tienden a derrapar y patinar, por lo que la odometría sólo es precisa en distancias cortas. Las fuerzas externas, como el viento y las corrientes oceánicas, aumentan la incertidumbre posicional. Los sensores inerciales, como los giroscopios, reducen la incertidumbre al confiar en la resistencia de la masa al cambio de velocidad.

Otros aspectos importantes del estado del robot se miden mediante sensores de fuerza y sensores de torsión. Son indispensables cuando los robots manipulan objetos frágiles o cuyo tamaño y forma exactos se desconocen. Imagine un manipulador robótico de una tonelada enroscando una bombilla. Sería muy fácil aplicar demasiada fuerza y romper la bombilla. Los sensores de fuerza le permiten al robot sentir con qué fuerza agarra la bombilla y los sensores de torsión le permiten sentir con qué fuerza gira. Los sensores de alta calidad pueden medir fuerzas en las tres direcciones de traslación y tres direcciones de rotación. Lo hacen a una frecuencia de varios cientos de veces por segundo para que un robot pueda detectar rápidamente fuerzas inesperadas y corregir sus acciones antes de que se rompa una bombilla. Sin embargo, puede ser un desafío equipar un robot con sensores de alta gama y la potencia computacional para monitorearlos.

### Producindo movimiento

El mecanismo que inicia el movimiento de un efecto se llama actuador; los ejemplos incluyen transmisiones, engranajes, cables y varillajes. El tipo más común de actuador es el **actuador eléctrico**, que utiliza electricidad para hacer girar un motor. Se utilizan predominantemente en sistemas que necesitan movimiento de rotación, como las articulaciones de un brazo robótico. Los **actuadores hidráulicos** utilizan fluido hidráulico presurizado (como aceite o agua) y los **actuadores neumáticos** utilizan aire comprimido para generar movimiento mecánico.

Los actuadores se utilizan a menudo para mover juntas que conectan cuerpos rígidos (enlaces). Los brazos y las piernas tienen este tipo de articulaciones. En las articulaciones de revolución, un eslabón gira con respecto al otro. En las uniones prismáticas, un eslabón se desliza a lo largo del otro. Ambas son articulaciones de un solo eje (un eje de movimiento). Otros tipos de articulaciones incluyen articulaciones esféricas, cilíndricas y planas, que son articulaciones multieje.

Para interactuar con los objetos del entorno, los robots utilizan pinzas. El tipo más básico de pinza es la pinza de mandíbula paralela, con dos dedos y un único actuador que junta los dedos para agarrar objetos. Este efecto es amado y odiado por su simplicidad.

#### Shadow Hand & Glove for Dextrous Manipulation | Shadow Robot

Las pinzas de tres dedos ofrecen un poco más de flexibilidad manteniendo la simplicidad. En el otro extremo del espectro están las manos humanoides (antropomórficas). Por ejemplo, Shadow Dexterous Hand tiene un total de 20 actuadores. Esto ofrece mucha más flexibilidad para manipulaciones complejas, incluidas maniobras con manipuladores en la mano (piense en levantar su teléfono celular y girarlo en la mano para orientarlo hacia arriba), pero esta flexibilidad tiene un precio: aprender a controlarlas. Estas pinzas complejas son más desafiantes.

### ¿Qué tipo de problema resuelve la robótica?

Ahora que sabemos cuál podría ser el hardware del robot, estamos listos para considerar el software agente que impulsa el hardware para lograr nuestros objetivos. Primero debemos decidir el marco computacional para este agente.

Ya hemos señalado que los problemas de la robótica no son deterministas, parcialmente observables y multiagentes. Podemos ver que a veces los agentes cooperan y otras son competitivos. En un pasillo estrecho donde sólo un agente puede pasar primero, un robot y una persona colaboran porque ambos quieren asegurarse de no chocar entre sí. Pero en algunos casos pueden competir un poco para llegar rápidamente a su destino. Si el robot es demasiado educado y siempre deja espacio, puede quedarse atrapado en situaciones abarrotadas y nunca alcanzar su objetivo.

¿Cuál es la función de recompensa del robot en esta formulación? Por lo general, el robot actúa al servicio de un ser humano; por ejemplo, entregando una comida a un paciente del hospital para obtener una recompensa del paciente, no la suya propia. Para la mayoría de entornos de robótica, aunque los diseñadores de robots podrían intentar especificar una función de recompensa suficientemente buena, la verdadera función de recompensa recae en el usuario a quien se supone que el robot debe ayudar. El robot necesitará descifrar los deseos del usuario o confiar en un ingeniero para especificar una aproximación a los deseos del usuario.

En cuanto a los espacios de acción, estado y observación del robot, la forma más general es que las observaciones son señales sin procesar de los sensores (por ejemplo, las imágenes provenientes de las cámaras o los impactos del láser provenientes del lidar); las acciones son corrientes eléctricas brutas que se envían a los motores; y el estado es lo que el robot necesita saber para tomar decisiones. Esto significa que existe una enorme brecha entre las percepciones de bajo nivel y los controles motores, y los planes de alto nivel que el robot necesita hacer. Para cerrar la brecha, los robóticos desacoplan aspectos del problema para simplificarlo.

En robótica solemos utilizar una jerarquía de tres niveles. El nivel de **planificación de tareas** decide un plan o política para acciones de alto nivel, a veces llamadas primitivas de acción o submetas: moverse hacia la puerta, abrirla, ir al ascensor, presionar el botón, etc. Luego, la **planificación de movimientos** se encarga de encontrar un camino que lleva al robot de un punto a otro, logrando cada subobjetivo. Finalmente, el **control** se utiliza para lograr el movimiento planificado utilizando los actuadores del robot. Dado que el nivel de planificación de tareas normalmente se define sobre estados y acciones discretos, en este capítulo nos centraremos principalmente en la planificación y el control del movimiento.

Por otra parte, el aprendizaje de preferencias se encarga de estimar el objetivo de un usuario final y la predicción de personas se utiliza para pronosticar las acciones de otras personas en el entorno del robot. Todos estos se combinan para determinar el comportamiento del robot.

Siempre que dividimos un problema en partes separadas, reducimos la complejidad, pero renunciamos a oportunidades para que las partes se ayuden entre sí. La acción puede ayudar a mejorar la percepción y también a determinar qué tipo de percepción es útil. De manera similar, las decisiones a nivel de movimiento podrían no ser las mejores a la hora de tener en cuenta cómo se dará seguimiento a ese movimiento; o las decisiones a nivel de tarea podrían hacer que el plan de tarea no sea instanciable a nivel de movimiento. Entonces, con el progreso en estas áreas separadas viene el impulso para reintegrarlas: planificar y controlar el movimiento juntos, planificar tareas y movimientos juntos, y reintegrar la percepción, la predicción y la acción, cerrando el ciclo de retroalimentación.

Hoy en día, la robótica consiste en seguir progresando en cada área y, al mismo tiempo, aprovechar ese progreso para lograr una mejor integración.

## Percepción robótica

La percepción es el proceso mediante el cual los robots transforman las mediciones de los sensores en representaciones internas del entorno. Pero la percepción de la robótica debe lidiar con sensores adicionales como lidar y sensores táctiles.

La percepción es difícil porque los sensores son ruidosos y el entorno es parcialmente observable, impredecible y a menudo dinámico. En otras palabras, los robots tienen todos los problemas de estimación (o filtrado) de estado. Como regla general, una buena representación interna de un robot tiene tres propiedades:

1. Contienen suficiente información para que el robot tome buenas decisiones.
2. Están estructurados para que puedan actualizarse de manera eficiente.
3. Son naturales en el sentido de que las variables internas corresponden a variables de estado natural en el mundo físico.

Los [filtros de Kalman](#), los [HMM](#) y las [redes de Bayes](#) pueden representar los modelos de transición y de sensor de un entorno parcialmente observable, y describimos algoritmos exactos y aproximados para actualizar el estado de creencia: la distribución de probabilidad posterior sobre el entorno. Variables de estado. Para problemas de robótica, incluimos las acciones pasadas del propio robot como variables observadas en el modelo.

### Localización y mapeo

La localización es el problema de descubrir dónde están las cosas, incluido el propio robot. Para simplificar las cosas, consideremos un robot móvil que se mueve lentamente en un mundo plano bidimensional. Supongamos también que al robot se le proporciona un mapa exacto del entorno. La pose de dicho robot móvil se define por sus dos coordenadas cartesianas con valores **x** e **y** y su **rumbo** con valor **θ**.

En la aproximación cinemática, cada acción consiste en la especificación “instantánea” de dos velocidades: una velocidad de traslación **vt** y una velocidad de rotación **ωt**. Para intervalos de tiempo pequeños **Δt**, un modelo determinista crudo del movimiento de dichos robots viene dado por la notación  $\hat{\mathbf{X}}$  se refiere a una predicción de estado determinista. Por supuesto, los robots físicos son algo impredecibles. Esta distribución de probabilidad es el modelo de movimiento del robot. Modela los efectos del movimiento en la ubicación del robot.

A continuación, necesitamos un modelo de sensor. Consideraremos dos tipos de modelos de sensores. El primero supone que los sensores detectan características estables y reconocibles del entorno llamadas puntos de referencia. Para cada punto de referencia, se informa el alcance y el rumbo. Sin ruido, se puede calcular una predicción del alcance y el rumbo mediante geometría simple. Una vez más, el ruido distorsiona nuestras mediciones. Para simplificar las cosas, supongamos ruido gaussiano con covarianza, lo que nos da el modelo del sensor.

Se utiliza un modelo de sensor algo diferente para una matriz de sensores de alcance, cada uno de los cuales tiene un rumbo fijo con respecto al robot. Estos sensores producen un vector de valores. Dada una pose **xt**, el rango calculado a lo largo de la dirección del haz desde **xt** hasta el obstáculo más cercano. Como antes, esto se verá corrompido por el ruido gaussiano. Normalmente, asumimos que los errores para las diferentes direcciones del haz son independientes y están distribuidos de manera idéntica.

El filtro de Kalman, que representa el estado de creencia como un gaussiano multivariado único, y el filtro de partículas, que representa el estado de creencia mediante una colección de partículas que corresponden a estados. La mayoría de los algoritmos de localización modernos utilizan una de estas dos representaciones de la creencia del robot.

La localización mediante filtrado de partículas se denomina [localización de Monte Carlo](#) o MCL. Todo lo que tenemos que hacer es proporcionar el modelo de movimiento y el modelo de sensor adecuados. El funcionamiento del algoritmo se ve más abajo. Cuando el robot descubre dónde se encuentra dentro de un edificio de oficinas. En la primera imagen, las partículas están distribuidas uniformemente según la anterior, lo que indica incertidumbre global sobre la posición del robot. En la segunda imagen, llega el primer conjunto de mediciones y las partículas forman cúmulos en las zonas de alta creencia posterior. En el tercero, se dispone de suficientes mediciones para empujar todas las partículas a un solo lugar.

### Adaptive Monte Carlo Localization - Robotics Knowledgebase

El filtro de Kalman es la otra forma importante de localización. A medida que el robot se mueve, la incertidumbre en la estimación de su ubicación aumenta. Su error disminuye a medida que detecta el alcance y el rumbo hacia un punto de referencia con una ubicación conocida y aumenta nuevamente cuando el robot pierde de vista el punto de referencia. Los algoritmos EKF funcionan bien si los puntos de referencia se identifican fácilmente.

En algunas situaciones, no hay ningún mapa del entorno disponible. Entonces el robot deberá adquirir un mapa. Este es un problema del huevo y la gallina: el robot de navegación tendrá que determinar su ubicación en relación con un mapa que no conoce del todo y, al mismo tiempo, construir este mapa aunque no conozca su ubicación real. Este problema es importante para muchas aplicaciones de robots y se ha estudiado ampliamente bajo el nombre de localización y mapeo simultáneos, abreviado como **SLAM**.

Los problemas de **SLAM** se resuelven utilizando muchas técnicas probabilísticas diferentes, incluido el filtro de Kalman extendido discutido anteriormente. Usar el EKF es sencillo: simplemente aumente el vector de estado para incluir las ubicaciones de los puntos de referencia en el entorno. Afortunadamente, la actualización de EKF escala cuadráticamente, por lo que para mapas pequeños (por ejemplo, unos pocos cientos de puntos de referencia) el cálculo es bastante factible. A menudo se obtienen mapas más ricos utilizando métodos de relajación de gráficos, similares a las técnicas de inferencia de redes bayesianas.

#### Otros tipos de percepción

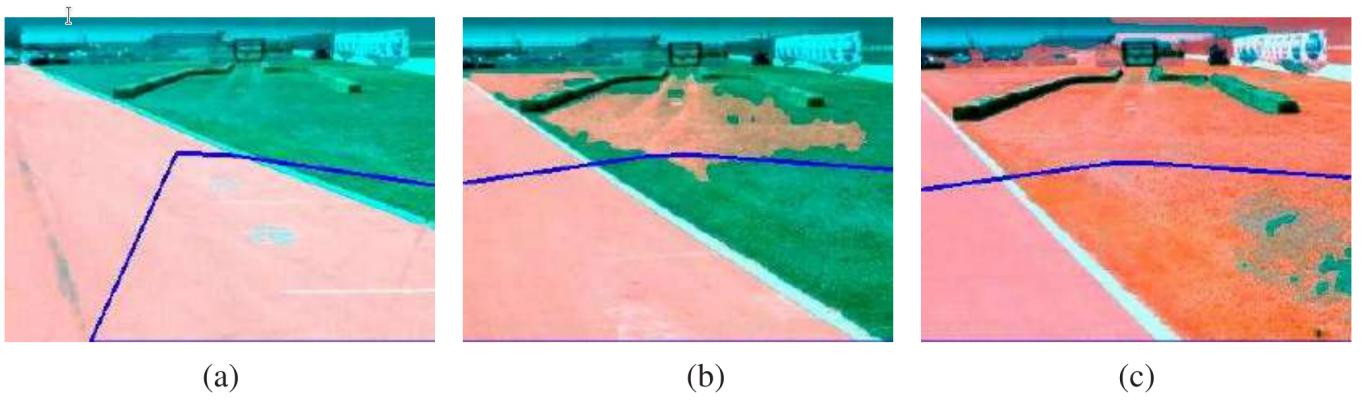
No toda la percepción de los robots tiene que ver con la localización o el mapeo. Los robots también perciben temperatura, olores, sonidos, etc. Muchas de estas cantidades pueden estimarse utilizando variantes de redes dinámicas de Bayes. Todo lo que se requiere para tales estimadores son distribuciones de probabilidad condicional que caractericen la evolución de las variables de estado a lo largo del tiempo y modelos de sensores que describan la relación de las mediciones con las variables de estado.

La tendencia en robótica es claramente hacia representaciones con una semántica bien definida. Las técnicas probabilísticas superan a otros enfoques en muchos problemas de percepción difíciles, como la localización y el mapeo. Sin embargo, las técnicas estadísticas son a veces demasiado engorrosas y soluciones más simples pueden ser igual de efectivas en la práctica.

#### Aprendizaje supervisado y no supervisado en percepción de robots

El aprendizaje automático juega un papel importante en la percepción de los robots. Este es particularmente el caso cuando no se conoce cuál es la mejor representación interna. Un enfoque común es mapear flujos de sensores de alta dimensión en espacios de menor dimensión utilizando métodos de aprendizaje automático no supervisados. Este enfoque se denomina incrustación de baja dimensión. El aprendizaje automático permite aprender modelos de sensores y de movimiento a partir de datos y, al mismo tiempo, descubrir una representación interna adecuada.

Otra técnica de aprendizaje automático permite a los robots adaptarse continuamente a grandes cambios en las mediciones de los sensores. Imagínese caminando desde un espacio iluminado por el sol hacia una habitación oscura con luces de neón. Claramente, las cosas son más oscuras por dentro. Pero el cambio de fuente de luz también afecta a todos los colores: la luz de neón tiene un componente de luz verde más fuerte que la luz solar. Sin embargo, de alguna manera parece que no notamos el cambio. Si entramos con personas en una habitación iluminada con luces de neón, no creemos que sus caras de repente se pongan verdes. Nuestra percepción se adapta rápidamente a las nuevas condiciones de iluminación y nuestro cerebro ignora las diferencias.



Las técnicas de percepción adaptativa permiten a los robots adaptarse a tales cambios. En la Figura superior se muestra un ejemplo, tomado del ámbito de la conducción autónoma. Aquí un vehículo terrestre no tripulado adapta su clasificador del concepto "superficie transitable". ¿Cómo funciona esto? El robot utiliza un láser para clasificar un área pequeña inmediatamente frente al robot. Cuando en el escaneo del alcance del láser se encuentra que esta zona es plana, se utiliza como ejemplo de entrenamiento positivo para el concepto "superficie transitable". Luego se entrena una técnica de mezcla de gaussianos, las imágenes anteriores son el resultado de aplicar este clasificador a la imagen completa.

Los métodos que hacen que los robots recopilen sus propios datos de entrenamiento (¡con etiquetas!) se denominan autosupervisados. En este caso, el robot utiliza el aprendizaje automático para aprovechar un sensor de corto alcance que funciona bien para la clasificación del terreno y convertirlo en un sensor que puede ver mucho más lejos. Eso permite que el robot conduzca más rápido y desacelere solo cuando el modelo del sensor indica que hay un cambio en el terreno que debe ser examinado más cuidadosamente por los sensores de corto alcance.

## Planificación y Control

En última instancia, las deliberaciones del robot se reducen a decidir cómo moverse, desde el nivel de la tarea abstracta hasta las corrientes que se envían a sus motores.

Empezamos separando el movimiento del control. Definimos una trayectoria como una secuencia de puntos en el espacio geométrico que seguirá un robot (o una parte de un robot, como un brazo). Esto está relacionado con la noción de camino, pero aquí nos referimos a una secuencia de puntos en el espacio más que a una secuencia de acciones discretas. La tarea de encontrar un buen camino se llama planificación del movimiento.

Una vez que tenemos un camino, la tarea de ejecutar una secuencia de acciones para seguir el camino se llama control de seguimiento de trayectoria. Una trayectoria es un camino que tiene un tiempo asociado con cada punto del camino. Un camino simplemente dice "ir de A a B, a C, etc." y una trayectoria dice "empieza en A, toma 1 segundo para llegar a B y otros 1,5 segundos para llegar a C, etc."

### Espacio de configuración

Un espacio de configuración, se refiere a un espacio abstracto que representa todas las posibles configuraciones o estados que un sistema puede adoptar. Esto es especialmente relevante en el contexto de sistemas autónomos como robots o agentes inteligentes, donde el sistema necesita tomar decisiones sobre qué acción tomar en función de su entorno y su estado interno.

En el caso de los robots, el espacio de configuración podría incluir variables como la posición y orientación del robot, las posibles velocidades y aceleraciones que puede alcanzar, las posiciones de los objetos en su entorno, entre otros factores relevantes para la tarea que debe realizar el robot.

Entender y explorar el espacio de configuración es fundamental para diseñar algoritmos y sistemas que puedan tomar decisiones efectivas y eficientes en diferentes situaciones.

### Planificación de movimiento

El problema de la planificación del movimiento consiste en encontrar un plan que lleve a un robot de una configuración a otra sin chocar con un obstáculo. Es un componente básico para el movimiento y la manipulación.

El problema de planificación del movimiento a veces se denomina problema de la mudanza del piano. Debe su nombre a los esfuerzos de una empresa de mudanzas por llevar un piano grande y de forma irregular de una habitación a otra sin golpear nada. Se nos da:

- un mundo de espacio de trabajo  $W$  (World),
- una región de obstáculo  $O \subset W$  (Obstacle),
- un robot con un espacio de configuración  $C$  (Configuration) y un conjunto de puntos  $A(q)$  para  $q \in C$ ,
- una configuración inicial  $qs \in C$ , y ( $q$  start)
- una configuración objetivo  $qg \in C$ . ( $q$  goal)

La región del obstáculo induce un obstáculo en el espacio y su correspondiente espacio libre definido como en la sección anterior. Necesitamos encontrar un camino continuo a través del espacio libre. Usaremos una curva parametrizada, para representar el camino.

El problema de la planificación del movimiento puede hacerse más complejo de varias maneras: definiendo el objetivo como un conjunto de configuraciones posibles en lugar de una configuración única; definir una función de costo (por ejemplo, longitud del camino) a minimizar; satisfacer las limitaciones (por ejemplo, si el camino implica llevar una taza de café, asegurarse de que la taza esté siempre orientada en posición vertical para que el café no se derrame).

Ahora consideremos algunas formas de resolver el problema de planificación del movimiento.

#### GRÁFICOS DE VISIBILIDAD

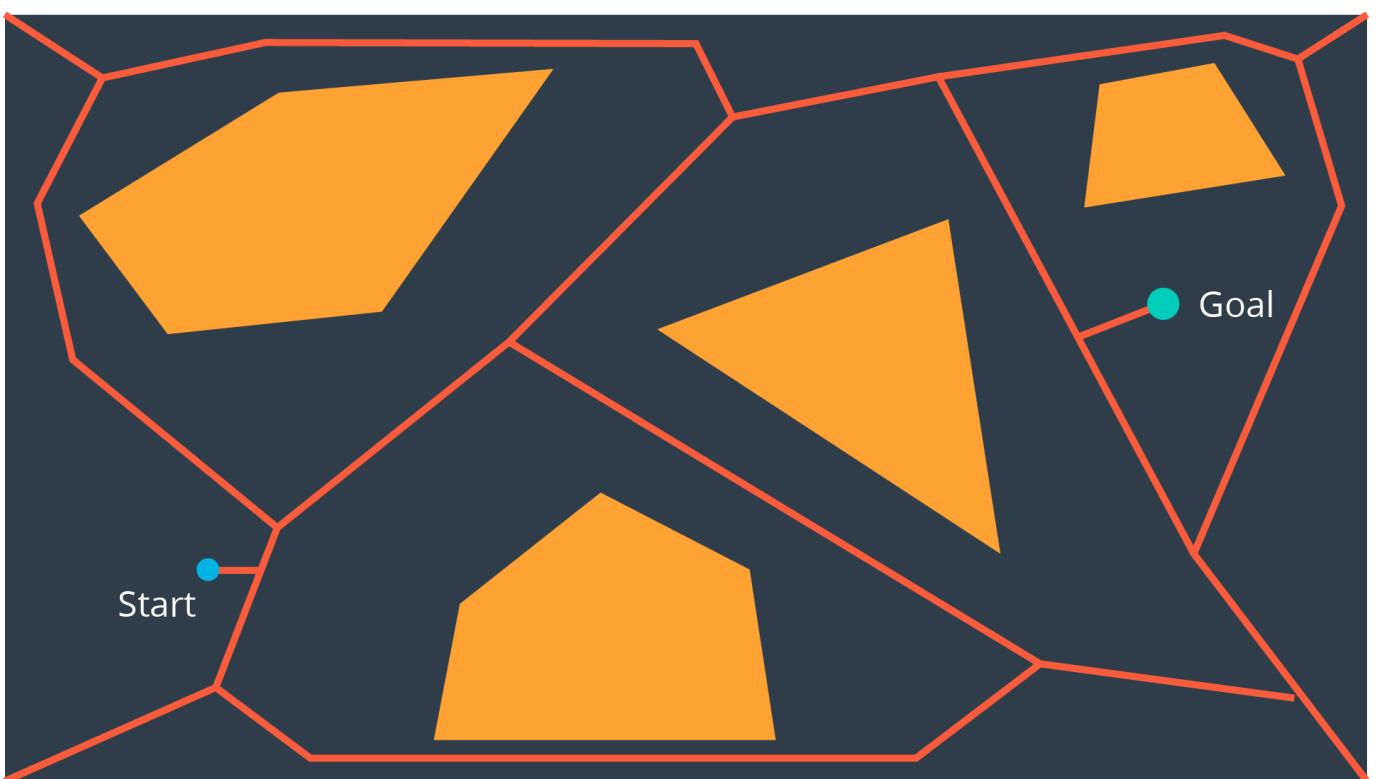
##### Gráficos de visibilidad

Los gráficos de visibilidad son una técnica comúnmente utilizada en la planificación de movimiento para robots móviles. Estos gráficos representan una estructura de datos que modela la conectividad entre diferentes puntos en un entorno, teniendo en cuenta las líneas de visibilidad entre estos puntos. Son particularmente útiles en entornos con obstáculos donde se desea encontrar un camino libre de colisiones para un robot.

Se utilizan del siguiente modo:

1. **Representación del entorno:** Para crear un gráfico de visibilidad, primero se necesita una representación del entorno en el que se moverá el robot. Esta representación puede ser en forma de un mapa discretizado, donde los obstáculos se representan como áreas intransitables, o puede ser en forma de una nube de puntos que representa los límites de los obstáculos.
2. **Puntos de visibilidad:** Se identifican los puntos en el espacio que tienen líneas de visibilidad claras entre ellos. Estos puntos generalmente incluyen vértices de los obstáculos y puntos de intersección de líneas de visión claras en el espacio libre entre los obstáculos.
3. **Construcción del grafo:** Se construye un grafo donde los nodos representan los puntos de visibilidad y las aristas representan las líneas de visión claras entre ellos. Es importante tener en cuenta que este grafo puede ser dirigido o no dirigido, dependiendo de la aplicación específica.
4. **Algoritmos de búsqueda:** Una vez que se ha construido el grafo de visibilidad, se pueden utilizar algoritmos de búsqueda como A\* o Dijkstra para encontrar el camino más corto y seguro entre dos puntos en el entorno. Estos algoritmos operan en el grafo de visibilidad y tienen en cuenta las restricciones de movimiento del robot y la presencia de obstáculos.
5. **Optimización y refinamiento:** A menudo, los caminos encontrados por los algoritmos de búsqueda pueden ser subóptimos o no factibles debido a limitaciones del entorno o del robot. En este caso, se pueden aplicar técnicas de optimización y refinamiento para mejorar el camino encontrado, como suavizado de trayectorias o replanificación dinámica.

DIAGRAMAS DE VORONOI



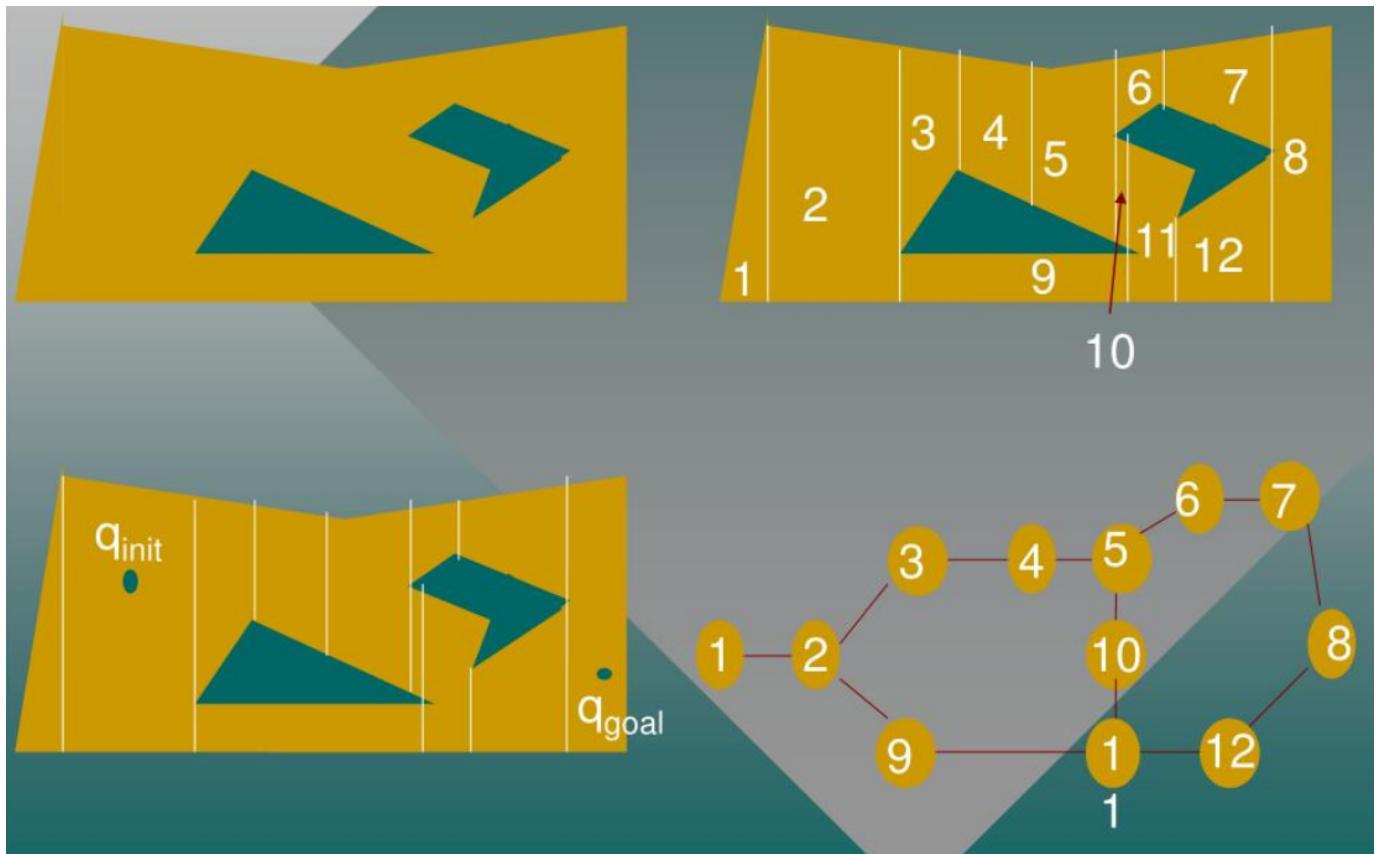
Estos gráficos dividen un espacio en regiones que representan el área más cercana a un conjunto dado de puntos de control o sitios. En el contexto de la planificación de movimiento para robots, los puntos de control suelen ser los obstáculos en el entorno.

La forma de utilizarlos es:

1. **Identificación de puntos de control:** los puntos de control suelen ser los límites de los obstáculos en el entorno. Estos puntos pueden ser esquinas, vértices o cualquier punto distintivo en los límites de los obstáculos.
2. **Diagrama de Voronoi:** Una vez que se han identificado los puntos de control, se construye el diagrama de Voronoi. Este diagrama divide el espacio en regiones donde cada región está asociada con uno de los puntos de control y contiene todos los puntos que están más cerca de ese punto de control que de cualquier otro. Estas regiones se llaman celdas de Voronoi.
3. **Gráfico de Voronoi:** Las celdas de Voronoi se representan como polígonos, y los límites entre las celdas se representan como líneas. Este gráfico proporciona una visualización clara de cómo se divide el espacio en función de la proximidad a los puntos de control.
4. **Planificación de trayectorias:** Una vez que se ha construido el gráfico de Voronoi, se puede utilizar para planificar trayectorias seguras para el robot móvil. Esto se logra encontrando un camino que minimice la distancia al punto de control más cercano en cada punto de la trayectoria, lo que garantiza que el robot se mantenga lo más alejado posible de los obstáculos.

**5. Optimización y refinamiento:** A menudo, los caminos encontrados utilizando el gráfico de Voronoi pueden ser subóptimos o no factibles debido a restricciones adicionales del entorno o del robot. En estos casos, se pueden aplicar técnicas de optimización y refinamiento para mejorar el camino encontrado, como suavizado de trayectorias o replanificación dinámica.

## DECOMPOSICIÓN CELULAR



La descomposición celular (o cellular decomposition, en inglés) se basa en dividir el espacio del entorno en regiones más simples y manejables, lo que facilita la planificación de trayectorias para el robot.

Procedemos de la siguiente manera:

- Representación del entorno:** Para utilizar la descomposición celular, primero necesitas una representación del entorno en el que el robot se moverá. Esto podría ser un mapa discreto donde los obstáculos se representan como áreas intransitables, o podría ser una nube de puntos que represente los límites de los obstáculos.
- División del espacio:** La descomposición celular divide el espacio del entorno en celdas más pequeñas y manejables. Estas celdas pueden tener diferentes formas y tamaños dependiendo del enfoque específico de la descomposición utilizada. Una opción común es dividir el espacio en celdas regulares (por ejemplo, cuadrados en un plano 2D o cubos en un espacio 3D).
- Conexiones entre celdas:** Una vez que se han definido las celdas, se determinan las conexiones entre ellas. Esto implica identificar qué celdas son adyacentes entre sí y, por lo tanto, podrían ser transitables para el robot. Las conexiones pueden basarse en la geometría del entorno o en otros criterios relevantes para la planificación de movimiento, como la distancia o la visibilidad entre celdas.
- Creación de un grafo de movimiento:** Con las celdas y las conexiones definidas, se crea un grafo donde los nodos representan las celdas y las aristas representan las conexiones entre ellas. Este grafo se utiliza entonces para encontrar una trayectoria libre de colisiones para el robot, utilizando algoritmos de búsqueda como A\* o Dijkstra.
- Optimización y refinamiento:** Una vez que se ha encontrado una trayectoria utilizando el grafo de movimiento, es posible que se requiera optimización y refinamiento adicionales para mejorar la calidad de la trayectoria. Esto podría implicar técnicas como suavizado de trayectorias, replanificación dinámica o tener en cuenta restricciones específicas del robot o del entorno.

## PLANIFICACIÓN DE MOVIMIENTO ALEATORIA (PROBABILISTIC ROADMAP PRM)

Se basa en la generación y evaluación de trayectorias de manera aleatoria en el espacio de configuración del robot. Aunque puede sonar simple, esta técnica puede ser sorprendentemente efectiva en entornos complejos o desconocidos donde no es posible utilizar métodos deterministas debido a la falta de información precisa sobre el entorno o la presencia de obstáculos dinámicos.

Aquí hay una explicación más detallada de cómo funciona la planificación de movimiento aleatoria:

1. **Inicialización:** La planificación de movimiento aleatoria comienza con la inicialización de una trayectoria de movimiento para el robot. Esto generalmente implica definir un punto de partida y un objetivo dentro del espacio de trabajo del robot.
2. **Generación de trayectorias aleatorias:** A partir del punto de partida, se generan aleatoriamente una serie de posibles trayectorias de movimiento para el robot. Estas trayectorias pueden ser generadas de varias formas, como seleccionar puntos aleatorios dentro del espacio de trabajo y trazar una trayectoria entre ellos, o generando secuencias aleatorias de movimientos elementales que el robot puede realizar.
3. **Evaluación de las trayectorias:** Cada trayectoria generada aleatoriamente se evalúa para determinar su calidad en función de ciertos criterios. Estos criterios pueden incluir la longitud de la trayectoria, la cantidad de colisiones con obstáculos, la distancia al objetivo, entre otros. Se pueden utilizar heurísticas o funciones de coste para asignar una puntuación a cada trayectoria en función de estos criterios.
4. **Selección de la mejor trayectoria:** Después de evaluar todas las trayectorias generadas aleatoriamente, se selecciona la mejor trayectoria según los criterios establecidos. Esta trayectoria puede no ser la óptima, pero se considera satisfactoria dadas las limitaciones del enfoque aleatorio.
5. **Refinamiento y repetición:** Una vez que se ha seleccionado una trayectoria, se puede realizar un refinamiento adicional para mejorar su calidad. Esto puede implicar suavizar la trayectoria, resolver colisiones potenciales o realizar ajustes para adaptarse mejor al entorno. Además, si la trayectoria no es satisfactoria, el proceso puede repetirse generando y evaluando nuevas trayectorias aleatorias hasta que se encuentre una solución aceptable.

#### ÁRBOLES ALEATORIOS QUE SE EXPLORAN RÁPIDAMENTE

RRT (Rapidly-exploring Random Trees) es una variante del algoritmo busca generar un camino entre un punto inicial y un punto objetivo explorando desde ambos extremos simultáneamente. Aquí te explico cómo funciona:

1. **Inicialización:** Se inicia con dos árboles, uno que comienza en el punto inicial y otro que comienza en el punto objetivo. Cada árbol consta de un solo nodo que representa su respectivo punto de partida.
2. **Expansión aleatoria:** En cada iteración del algoritmo, se genera aleatoriamente un nuevo punto en el espacio de configuración, uno para cada árbol. Estos puntos aleatorios son generados de manera similar a como se describe en el RRT estándar, posiblemente influenciados por la posición del otro punto objetivo.
3. **Extensión de árboles:** Cada nuevo punto generado se conecta al nodo más cercano en el árbol correspondiente. Esto implica que cada árbol se expande hacia el punto generado. Se verifica la validez de las conexiones y se añaden los nuevos nodos al árbol si la conexión es viable.
4. **Colisión de árboles:** En cada iteración, se verifica si los árboles se han encontrado entre sí. Esto se puede hacer comprobando si un nuevo nodo generado en un árbol se encuentra dentro de un cierto radio de cercanía de un nodo en el otro árbol. Si se encuentran, se ha encontrado una posible solución.
5. **Construcción del camino:** Una vez que los dos árboles se han encontrado, se puede construir un camino entre el punto inicial y el punto objetivo. Esto se hace trazando una trayectoria desde el nodo del árbol inicial hasta el nodo del árbol objetivo, combinando las trayectorias de ambos árboles.
6. **Optimización y refinamiento:** Después de encontrar un camino inicial, se puede realizar una optimización adicional para mejorar su calidad. Esto puede implicar suavizar la trayectoria, resolver colisiones potenciales o eliminar trayectorias redundantes.
7. **Finalización y selección de la mejor trayectoria:** Una vez que se ha encontrado una trayectoria satisfactoria, se selecciona como la solución del problema de planificación de movimiento.

#### OPTIMIZACIÓN DE TRAYECTORIA PARA PLANIFICACIÓN CINEMÁTICA

La optimización de trayectoria en la planificación cinemática para robots se refiere al proceso de mejorar una trayectoria generada inicialmente para que cumpla con ciertos criterios específicos, como minimizar el tiempo de ejecución, reducir la energía consumida, evitar colisiones o maximizar la suavidad de la trayectoria. Aquí están los pasos principales involucrados en la optimización de trayectorias para la planificación de movimiento de robots:

1. **Definición de la función objetivo:** Antes de iniciar la optimización, es necesario definir una función objetivo que capture los objetivos de optimización deseados. Esta función puede tener varios componentes, como la distancia recorrida, el tiempo de ejecución, la energía consumida, la suavidad de la trayectoria o la distancia mínima a los obstáculos.
2. **Formulación del problema de optimización:** Una vez que se ha definido la función objetivo, el problema de optimización se formula para encontrar la trayectoria que minimiza o maximiza esta función objetivo, sujeto a ciertas restricciones. Estas restricciones pueden incluir limitaciones en la velocidad, aceleración o jerarquías cinemáticas del robot, así como restricciones de colisión con el entorno.
3. **Selección de algoritmo de optimización:** Existen diversos algoritmos de optimización que pueden utilizarse para resolver el problema formulado. Algunos de los algoritmos comúnmente utilizados incluyen el método del gradiente descendente, algoritmos de búsqueda heurística como algoritmos genéticos, optimización basada en enjambres de partículas (PSO), optimización por enjambre de hormigas (ACO), entre otros.

4. **Aplicación del algoritmo de optimización:** Una vez seleccionado el algoritmo adecuado, se aplica para encontrar la trayectoria que optimiza la función objetivo. Esto implica ejecutar el algoritmo en iteraciones sucesivas, donde en cada iteración se ajusta la trayectoria actual en función de la función objetivo y las restricciones.
5. **Evaluación de la solución obtenida:** Despues de que el algoritmo de optimización converge o alcanza un cierto criterio de finalización, se evalúa la solución obtenida para asegurarse de que cumple con los requisitos deseados. Esto puede implicar la simulación de la trayectoria planificada en un entorno virtual para verificar la ausencia de colisiones y la suavidad de la trayectoria.
6. **Refinamiento y ajuste:** Si es necesario, se pueden realizar ajustes adicionales en la trayectoria obtenida para mejorar su calidad. Esto puede incluir técnicas de suavizado de trayectorias para reducir las discontinuidades, ajustes locales para evitar colisiones o cambios en la parametrización de la trayectoria para optimizarla aún más.

#### **Control de seguimiento de trayectoria**

El control de seguimiento de trayectoria en el movimiento de robots se encarga de guiar al robot para que siga una trayectoria planificada lo más cercanamente posible, considerando limitaciones del sistema y perturbaciones externas. Este proceso implica varios pasos:

1. **Obtención de la trayectoria planificada:** Se adquiere la trayectoria deseada que se espera que el robot siga.
2. **Realimentación de la información del sistema:** Se recopila información en tiempo real sobre el estado del robot, como su posición, velocidad y orientación.
3. **Comparación con la trayectoria planificada:** Se compara continuamente el estado actual del robot con la trayectoria planificada para determinar el error de seguimiento.
4. **Diseño del controlador:** Se diseña un controlador que genere comandos de control para minimizar el error y guiar al robot hacia la trayectoria deseada.
5. **Aplicación de los comandos de control:** Los comandos de control se aplican al sistema del robot para influir en su movimiento y lograr que siga la trayectoria planificada.
6. **Realimentación y ajuste continuo:** Se ajustan los comandos de control en cada ciclo de control para mantener al robot en la trayectoria deseada, incluso en presencia de perturbaciones.

Para lograr este seguimiento de trayectoria, se utilizan diferentes tipos de controladores, como:

- **Controlador Proporcional (P):** Aplica fuerza en proporción negativa al error observado entre la posición real y deseada del robot.
- **Controlador Proporcional-Derivado (PD):** Extiende el control proporcional al agregar un término que es proporcional a la primera derivada del error a lo largo del tiempo, lo que amortigua el sistema controlado.
- **Controlador Proporcional-Integral-Derivado (PID):** Incluye un término adicional que integra el error en el tiempo, lo que ayuda a corregir errores sistemáticos prolongados.
- **Control de Par Calculado:** Combina la dinámica inversa con la realimentación del error para calcular el par necesario que el modelo del robot cree que se requiere, compensando la inexactitud del modelo con términos de error proporcionales.

#### PLANES VERSUS POLÍTICAS

Con el movimiento en robótica, en realidad estamos considerando un MDP (Markov Decision Process) subyacente donde los estados son estados dinámicos (configuración y velocidad) y las acciones son entradas de control, generalmente en forma de pares. Si echas otro vistazo a nuestras leyes de control anteriores, son políticas, no planes: le dicen al robot qué acción tomar desde cualquier estado al que pueda llegar. Sin embargo, suelen estar lejos de ser políticas óptimas. Debido a que el estado dinámico es continuo y de alta dimensión (al igual que el espacio de acción), las políticas óptimas son computacionalmente difíciles de extraer.

En cambio, lo que hicimos aquí fue solucionar el problema. Primero elaboramos un plan, en un estado y un espacio de acción simplificados: usamos solo el estado cinemático y asumimos que los estados son alcanzables entre sí sin prestar atención a la dinámica subyacente. Esta es la planificación del movimiento y nos da la ruta de referencia.

Pero como nuestro modelo dinámico suele ser erróneo, lo convertimos en una política que intenta seguir el plan, volviendo a él cuando se aleja. Al hacer esto, introducimos suboptimidad de dos maneras: primero, planificando sin considerar la dinámica y, segundo, asumiendo que si nos desviamos del plan, lo óptimo es volver al plan original. A continuación, describimos técnicas que calculan políticas directamente sobre el estado dinámico, evitando la separación por completo.

#### **Control óptimo**

En lugar de utilizar un planificador para crear una ruta cinemática y preocuparse únicamente por la dinámica del sistema después del hecho, aquí analizamos cómo podríamos hacerlo todo a la vez. Tomaremos el problema de optimización de

trayectorias para rutas cinemáticas y lo convertiremos en una verdadera optimización de trayectorias con dinámica: optimizaremos directamente sobre las acciones, teniendo en cuenta la dinámica (o transiciones).

El enfoque propuesto combina la planificación de trayectorias con la dinámica del sistema en una sola optimización. Se busca una secuencia de acciones que minimice un costo acumulado, teniendo en cuenta la transición del estado del sistema y las restricciones. Esto se relaciona con la planificación de movimiento y el control de seguimiento de trayectorias al considerar configuraciones y acciones simultáneamente.

Para resolver este problema, se pueden tomar gradientes del costo acumulado con respecto a las acciones. Se utilizan técnicas de optimización de trayectorias como el tiroteo múltiple y la colocación directa. Cuando el costo es cuadrático y la dinámica es lineal, se puede aplicar el regulador cuadrático lineal (LQR), que encuentra una política óptima de forma eficiente. Aunque los problemas reales rara vez cumplen estas condiciones, el LQR y su variante ILQR se utilizan ampliamente en la práctica.

## Planificación de movimientos inciertos

En robótica, la incertidumbre surge de la observabilidad parcial del entorno y de los efectos estocásticos (o no modelados) de las acciones del robot. También pueden surgir errores por el uso de algoritmos de aproximación, como el filtrado de partículas, que no le dan al robot un estado de creencia exacto incluso si el entorno está modelado perfectamente.

La mayoría de los robots actuales utilizan algoritmos deterministas para la toma de decisiones, como los algoritmos de planificación de ruta de la sección anterior o los algoritmos de búsqueda. Estos algoritmos deterministas se adaptan de dos maneras: en primer lugar, se ocupan de la espacio de estado continuo convirtiéndolo en un espacio discreto (por ejemplo, con gráficos de visibilidad o descomposición de celdas). En segundo lugar, abordan la incertidumbre en el estado actual eligiendo el estado más probable a partir de la distribución de probabilidad producida por el algoritmo de estimación del estado. Ese enfoque hace que el cálculo sea más rápido y se adapta mejor a los algoritmos de búsqueda deterministas. En esta sección analizamos métodos para abordar la incertidumbre.

1. **Replanificación en línea:** En entornos inciertos, los planes deterministas pueden volverse subóptimos. La replanificación en línea, como el control predictivo de modelos (MPC), permite recalcular continuamente planes basados en nueva información. Esto se logra planificando para un horizonte temporal corto y ajustando los planes en cada paso del tiempo.
2. **Acciones de recopilación de información:** La incertidumbre también requiere acciones específicas para recopilar información relevante. En lugar de separar la estimación del control, se puede resolver un Proceso de Decisión de Markov Parcialmente Observado (POMDP) para considerar la incertidumbre en la planificación. Esto permite que el robot razonne sobre la información futura que podría obtener y tome acciones óptimas considerando tanto la información actual como la futura.

## Aprendizaje por refuerzo en robótica

Hasta ahora hemos considerado tareas en las que el robot tiene acceso al modelo dinámico del mundo. En muchas tareas, es muy difícil escribir un modelo de este tipo, lo que nos sitúa en el dominio del aprendizaje por refuerzo (RL).

Un desafío de RL en robótica es la naturaleza continua de los espacios de estado y acción, que manejamos mediante discretización o, más comúnmente, mediante aproximación de funciones. Las políticas o funciones de valor se representan como combinaciones de características útiles conocidas o como redes neuronales profundas. Las redes neuronales pueden mapear desde entradas sin procesar directamente a salidas y, por lo tanto, evitan en gran medida la necesidad de ingeniería de características, pero requieren más datos.

Un desafío mayor es que los robots operen en el mundo real. Hemos visto cómo se puede utilizar el aprendizaje por refuerzo para aprender a jugar al ajedrez o al Go jugando juegos simulados. Pero cuando un robot real se mueve en el mundo real, tenemos que asegurarnos de que sus acciones sean seguras (¡las cosas se rompen!), y tenemos que aceptar que el progreso será más lento que en una simulación porque el mundo se niega a moverse más rápido que un segundo por segundo. Gran parte de lo interesante del uso del aprendizaje por refuerzo en robótica se reduce a cómo podemos reducir la complejidad de las muestras del mundo real: el número de interacciones con el mundo físico que el robot necesita antes de aprender a realizar la tarea.

## Humanos y robots

Hasta ahora, nos hemos centrado en la planificación de un robot y en aprender a actuar de forma aislada. Esto es útil para algunos robots, como los rovers que enviamos a explorar planetas distantes en nuestro nombre. Pero, en general, no construimos robots para que funcionen de forma aislada. Los construimos para ayudarnos y para trabajar en entornos humanos, a nuestro alrededor y con nosotros.

Esto plantea dos desafíos complementarios. El primero es optimizar la recompensa cuando hay personas actuando en el mismo entorno que el robot. A esto lo llamamos problema de coordinación. Cuando la recompensa del robot depende no sólo de sus propias acciones, sino también de las acciones que realizan las personas, el robot tiene que elegir sus acciones de una manera que combine bien con las de ellos. Cuando el humano y el robot están en el mismo equipo, esto se convierte en colaboración.

En segundo lugar está el desafío de optimizar lo que la gente realmente quiere. Si un robot va a ayudar a las personas, su función de recompensa debe incentivar las acciones que las personas quieren que ejecute el robot. Determinar la función (o política) de recompensa adecuada para el robot es en sí mismo un problema de interacción. Exploraremos estos dos desafíos uno por uno.

## Coordinación

Supongamos por ahora, como hasta ahora, que el robot tiene acceso a una función de recompensa claramente definida. Pero, en lugar de necesitar optimizarlo de forma aislada, ahora el robot necesita optimizarlo en torno a un humano que también actúa. Por ejemplo, cuando un automóvil autónomo se incorpora a la autopista, necesita negociar la maniobra con el conductor humano que viene al carril objetivo: ¿debería acelerar y incorporarse al frente, o reducir la velocidad y incorporarse a la parte trasera? Más tarde, cuando se detiene ante una señal de stop, preparándose para girar a la derecha, tiene que tener cuidado con el ciclista en el carril bici y con el peatón que está a punto de pisar el paso de peatones.

O considere un robot móvil en un pasillo. Alguien que se dirige directamente hacia el robot da un paso ligeramente hacia la derecha, indicando por qué lado del robot quiere pasar. El robot tiene que responder, aclarando sus intenciones.

### LOS HUMANOS COMO AGENTES APROXIMADAMENTE RACIONALES

Una forma de formular la coordinación con un humano es modelarla como un juego entre el robot y el humano. Con este enfoque, asumimos explícitamente que las personas son agentes incentivados por objetivos. Esto no significa automáticamente que sean agentes perfectamente racionales (es decir, que encuentren soluciones óptimas en el juego), pero sí significa que el robot puede estructurar la forma en que razona sobre el humano a través de la noción de posibles objetivos que el humano podría tener.

Tres aspectos importantes complican este juego. La **primera** es que el humano y el robot no necesariamente conocen los objetivos del otro. Esto lo convierte en un juego de información incompleta.

En **segundo** lugar, los espacios de estado y acción son continuos. Podemos hacer una búsqueda en árbol para abordar juegos discretos, pero ¿cómo abordamos espacios continuos?

En **tercer** lugar, aunque en el nivel alto el modelo de juego tiene sentido (los humanos se mueven y tienen objetivos), es posible que el comportamiento de un humano no siempre esté bien caracterizado como una solución al juego. El juego supone un desafío computacional no sólo para el robot, sino también para nosotros, los humanos. Requiere pensar en lo que hará el robot en respuesta a lo que hace la persona, lo cual depende de lo que el robot cree que hará la persona, y muy pronto llegamos a "¿qué crees que creo que creo que pienso?": son las tortugas. ¡toda la calle abajo! Los humanos no pueden lidiar con todo eso y exhiben ciertas subóptimas. Esto significa que el robot debe tener en cuenta estas subóptimas.

Entonces, ¿qué debe hacer un coche autónomo cuando el problema de coordinación es tan difícil? Tomaremos el juego y lo dividiremos en hacer predicciones sobre las acciones humanas y decidir qué debería hacer el robot dadas estas predicciones.

### PREDECIR LA ACCIÓN HUMANA

Predecir las acciones humanas es difícil porque dependen de las acciones del robot y viceversa. Un truco que utilizan los robots es fingir que la persona está ignorando al robot. El robot supone que las personas son óptimas con respecto a su objetivo, que el robot desconoce y que se modela como si ya no dependiera de las acciones del robot.

Así es como las acciones pasadas del humano acaban informando al robot sobre lo que el humano hará en el futuro. Tener una creencia sobre el objetivo del ser humano ayuda al robot a anticipar las próximas acciones que realizará el ser humano.

Lo mismo puede suceder al conducir. Puede que no sepamos cuánto valora otro conductor la eficiencia, pero si lo vemos acelerar cuando alguien intenta incorporarse delante de él, ahora sabemos un poco más sobre él. Y una vez que sepamos eso, podremos anticipar mejor lo que harán en el futuro: es probable que el mismo conductor se acerque más a nosotros o se abra paso entre el tráfico para adelantarnos.

### PREDICCIONES HUMANAS SOBRE EL ROBOT.

La información incompleta suele tener dos caras: el robot no conoce el objetivo del ser humano y el ser humano, a su vez, no conoce el objetivo del robot; es necesario que la gente haga predicciones sobre los robots. Como diseñadores de robots, no estamos a cargo de cómo el ser humano hace predicciones; sólo podemos controlar lo que hace el robot. Sin embargo, el robot puede actuar de manera que al humano le resulte más fácil hacer predicciones correctas. El robot puede suponer que el humano está usando algo más o menos análogo a la ecuación para estimar el objetivo del robot y, por lo tanto, el robot actuará de manera que su verdadero objetivo pueda inferirse fácilmente.

Un caso especial del juego es cuando el humano y el robot están en el mismo equipo, trabajando hacia la misma meta u objetivo. Imagínese tener un robot doméstico personal que lo ayude a preparar la cena o limpiar; estos son ejemplos de colaboración.

Ahora podemos definir un agente conjunto cuyas acciones son tuplas de acciones humano-robot y que optimiza para, y estamos resolviendo un problema de planificación habitual. Calculamos el plan o política óptima para el agente conjunto y listo, ahora sabemos qué deben hacer el robot y el humano.

Esto funcionaría muy bien si las personas fueran perfectamente óptimas. El robot haría su parte del plan conjunto, el humano la suya. Desafortunadamente, en la práctica, la gente no parece seguir el plan de agente conjunto perfectamente diseñado; ¡Tienen opinión propia! Sin embargo, ya hemos aprendido una forma de manejar esto con el control predictivo de modelo (MPC): la idea es idear un plan, ejecutar la primera acción y luego volver a planificar. De esta manera, el robot siempre adapta su plan a lo que realmente está haciendo el humano.

Supongamos que usted y el robot están en su cocina y han decidido hacer gofres. Estás un poco más cerca del frigorífico, por lo que el plan conjunto óptimo sería coger los huevos y la leche del frigorífico, mientras el robot recoge la harina del armario. El robot lo sabe porque puede medir con bastante precisión dónde está cada uno. Pero supongamos que empiezas a dirigirte al gabinete de harina. Estás yendo en contra del plan conjunto óptimo. En lugar de ceñirte a ello y obstinadamente ir también a por la harina, el robot recalcula el plan óptimo, y ahora que estás lo suficientemente cerca de la harina, lo mejor es que el robot agarre la plancha para gofres.

Si sabemos que las personas podrían desviarse del óptimo, podemos dar cuenta de ello con anticipación. El robot puede intentar anticipar que vas a por la harina en el momento en que das el primer paso (digamos, usando la técnica de predicción anterior). Aunque técnicamente sigue siendo óptimo que te des la vuelta y te dirijas al frigorífico, el robot no debería asumir que eso es lo que va a pasar. En cambio, el robot puede calcular un plan en el que usted sigue haciendo lo que parece querer.

### Aprender a hacer lo que los humanos quieren

Otra forma en que la interacción con los humanos entra en la robótica es en la propia JR: la función de costo o recompensa del robot. El marco de agentes racionales y los algoritmos asociados reducen el problema de generar un buen comportamiento a especificar una buena función de recompensa. Pero para los robots, como para muchos otros agentes de IA, todavía es difícil calcular correctamente el costo.

Tomemos como ejemplo los automóviles autónomos: queremos que lleguen al destino, que sean seguros, que conduzcan cómodamente para sus pasajeros, que obedezcan las leyes de tránsito, etc. Un diseñador de un sistema de este tipo necesita equilibrar estos diferentes componentes de la función de costos. La tarea del diseñador es difícil porque los robots están diseñados para ayudar a los usuarios finales y no todos los usuarios finales son iguales. Todos tenemos diferentes preferencias sobre la agresividad con la que queremos que conduzca nuestro coche, etc.

A continuación, exploraremos dos alternativas para intentar que el comportamiento del robot coincida con lo que realmente queremos que haga. La primera es aprender una función de costos a partir del aporte humano. La segunda es evitar la función de costos e imitar las demostraciones humanas de la tarea.

#### APRENDIZAJE PREFERENCIAL: FUNCIONES DE COSTO DE APRENDIZAJE

Imagine que un usuario final le muestra a un robot cómo realizar una tarea. Por ejemplo, conducen el coche de la forma que les gustaría que lo condujera el robot. ¿Se te ocurre alguna manera de que el robot utilice estas acciones (las llamamos "demostraciones") para determinar qué función de costos debería optimizar?

Si la persona conduce a la defensiva, la función de costos que explicará sus acciones pondrá mucho peso en la seguridad y menos en la eficiencia. El robot puede adoptar esta función de costos como propia y optimizarla cuando conduce el propio coche.

Hay otras formas en que una persona puede dar su opinión. Una persona podría utilizar el lenguaje en lugar de la demostración para instruir al robot. Una persona podría actuar como crítico, observando al robot realizar una tarea de una manera (o dos) y luego diciendo qué tan bien se hizo la tarea (o de qué manera fue mejor), o dando consejos sobre cómo mejorar.

#### POLÍTICAS DE APRENDIZAJE DIRECTAMENTE A TRAVÉS DE LA IMITACIÓN

Una alternativa es evitar las funciones de costos y aprender directamente la política de robot deseada. En nuestro ejemplo de automóvil, las demostraciones humanas generan un conveniente conjunto de datos de estados etiquetados por la acción que el robot debe realizar en cada estado. El robot puede ejecutar aprendizaje supervisado para ajustarse a una política, y ejecutar esa política. Esto se llama **aprendizaje por imitación** o **clonación conductual**.

Un desafío con este enfoque es la generalización a nuevos estados. El robot no sabe por qué las acciones en su base de datos han sido marcadas como óptimas. No tiene regla causal; todo lo que puede hacer es ejecutar un algoritmo de aprendizaje supervisado para intentar aprender una política que se generalizará a estados desconocidos. Sin embargo, no hay garantía de que la generalización sea correcta.

El robot puede ajustarse a un modelo dinámico basado en las demostraciones y luego utilizar un control óptimo para generar una política que optimice la permanencia cerca de la demostración. Se ha utilizado una versión de esto para realizar maniobras muy desafiantes a nivel experto en un pequeño helicóptero radiocontrolado.

Técnicas recientes relacionadas utilizan entrenamiento adversario: alternan entre entrenar a un clasificador para distinguir entre la política aprendida del robot y las demostraciones del humano, y entrenar una nueva política del robot mediante aprendizaje reforzado para engañar al clasificador. Estos avances permiten al robot manejar estados que están cerca de las demostraciones, pero la generalización a estados lejanos o a nuevas dinámicas es un trabajo en progreso.

## Dominios de aplicación

La tecnología robótica ya está impregnando nuestro mundo y tiene el potencial de mejorar nuestra independencia, salud y productividad. A continuación se muestran algunas aplicaciones de ejemplo.

**Cuidados en el hogar:** Los robots han comenzado a ingresar a los hogares para cuidar a los adultos mayores y a las personas con discapacidad motriz, ayudándolos con las actividades de la vida diaria y permitiéndoles vivir de manera más independiente. Estos incluyen sillas de ruedas y brazos montados en sillas de ruedas como el brazo Kinova. Aunque al principio son operados directamente por un humano, estos robots están ganando cada vez más autonomía. En el horizonte hay robots operados mediante interfaces cerebro-máquina, que se ha demostrado que permiten a las personas con cuadriplejia utilizar un brazo robótico para agarrar objetos e incluso alimentarse. Relacionados con esto están las prótesis que responden inteligentemente a nuestras acciones y los exoesqueletos que nos dan una fuerza sobrehumana o permiten que las personas que no pueden controlar sus músculos de la cintura para abajo vuelvan a caminar.

### Brazo Kinova

Los robots personales están destinados a ayudarnos con tareas diarias como limpiar y organizar, liberándonos tiempo. Aunque la manipulación todavía tiene un camino por recorrer antes de que pueda funcionar sin problemas en entornos humanos desordenados y desestructurados, la navegación ha logrado algunos avances. En particular, muchos hogares ya cuentan con un robot aspirador móvil.

### JASPER, robots aspiradores básicos - Robots al Detalle

**Atención médica:** los robots ayudan y potencian a los cirujanos, permitiendo procedimientos más precisos, mínimamente invasivos y seguros con mejores resultados para los pacientes. El robot quirúrgico Da Vinci ahora se utiliza ampliamente en hospitales de EE. UU.

### Robot Da Vinci para cirugía robótica. Especialidades y usos

**Servicios:** Los robots móviles ayudan en edificios de oficinas, hoteles y hospitales. Savioke ha instalado robots en hoteles que entregan productos como toallas o pasta de dientes en la habitación. Los robots Helpmate y TUG transportan alimentos y medicamentos en los hospitales, mientras que el robot Moxi de Diligent Robotics ayuda a las enfermeras con las responsabilidades logísticas de back-end. Co-Bot deambula por los pasillos de la Universidad Carnegie Mellon, listo para guiarte a la oficina de alguien. También podemos utilizar robots de telepresencia como el Beam para asistir a reuniones y conferencias de forma remota, robots de Telepresencia o controlar a nuestros abuelos.

**Automóviles autónomos:** algunos de nosotros ocasionalmente nos distraemos mientras conducimos, con llamadas de teléfono celular, mensajes de texto u otras distracciones. El triste resultado: más de un millón de personas mueren cada año en accidentes de tráfico. Además, muchos de nosotros pasamos mucho tiempo conduciendo y nos gustaría recuperar parte de ese tiempo. Todo esto ha llevado a un esfuerzo masivo y continuo para implementar automóviles autónomos.

**Entretenimiento:** Disney ha estado utilizando robots (bajo el nombre de animatronics) en sus parques desde 1963. Originalmente, estos robots estaban restringidos a movimientos (y habla) invariables, de circuito abierto y diseñados a mano, pero desde 2009 una versión llamada autonomatronics puede generar acciones autónomas. Los robots también toman la forma de juguetes inteligentes para niños; por ejemplo, Cozmo de Anki juega con niños y puede golpear la mesa con frustración cuando pierde. Finalmente, los cuadrotoros como el R1 de Skydio de la actúan como fotógrafos y camarógrafos personales, siguiéndonos para tomar fotografías de acción mientras esquiamos o andamos en bicicleta.

### Skydio R1 drone tracks and videos your ride - even through the trees

**Exploración y entornos peligrosos:** los robots han llegado a lugares donde ningún ser humano había llegado antes, incluida la superficie de Marte. Los brazos robóticos ayudan a los astronautas a desplegar y recuperar satélites y a construir la Estación Espacial Internacional. Los robots también ayudan a explorar bajo el mar. Se utilizan habitualmente para adquirir mapas de barcos hundidos. En 1996, un equipo de investigadores introdujo un robot con patas en el cráter de un volcán activo para adquirir datos para la investigación climática. Los robots se están convirtiendo en herramientas muy efectivas para recopilar información en dominios de difícil (o peligroso) acceso para las personas.

Los robots han ayudado a las personas a limpiar desechos nucleares, sobre todo en Three Mile Island, Chernobyl y Fukushima. Los robots estuvieron presentes después del colapso del World Trade Center, donde ingresaron a estructuras consideradas demasiado peligrosas para los equipos humanos de búsqueda y rescate. También en este caso, estos robots se implementan inicialmente mediante teleoperación y, a medida que avanza la tecnología, se vuelven cada vez más autónomos, con un operador humano a cargo pero sin tener que especificar cada comando.

**Industria:** la mayoría de los robots actuales se implementan en fábricas, automatizando tareas que son difíciles, peligrosas o aburridas para los humanos. (La mayoría de los robots industriales se encuentran en fábricas de automóviles). Automatizar estas tareas es positivo en términos de producir eficientemente lo que la sociedad necesita. Al mismo tiempo, también significa desplazar a algunos trabajadores humanos de sus puestos de trabajo. Esto tiene importantes implicaciones políticas y económicas: la necesidad de reentrenamiento y educación, la necesidad de una división justa de los recursos, etc.

## Resumen

---

La robótica se trata de agentes encarnados físicamente, que pueden cambiar el estado del mundo físico. En este capítulo, hemos aprendido lo siguiente:

- Los tipos de robots más comunes son los manipuladores (brazos robóticos) y los robots móviles. Tienen sensores para percibir el mundo y actuadores que producen movimiento, que luego afecta al mundo a través de efectores.
- El problema general de la robótica implica la estocasticidad (que puede ser manejada por los MDP), la observabilidad parcial (que puede ser manejada por los POMDP) y la actuación con y alrededor de otros agentes (que puede ser manejada con la teoría de juegos). El problema se complica aún más por el hecho de que la mayoría de los robots trabajan en espacios de acción y estados continuos y de alta dimensión. También operan en el mundo real, que se niega a correr más rápido que el tiempo real y en el que los fallos provocan daños a cosas reales, sin posibilidad de "deshacerlas".
- Idealmente, el robot resolvería todo el problema de una vez: las observaciones en forma de señales brutas de los sensores entran y las acciones en forma de pares o corrientes a los motores salen. Sin embargo, en la práctica esto resulta demasiado desalentador y los robóticos suelen desacoplar diferentes aspectos del problema y tratarlos de forma independiente.
- Normalmente separamos la percepción (estimación) de la acción (generación de movimiento). La percepción en robótica implica visión por computadora para reconocer el entorno a través de cámaras, pero también localización y mapeo.
- La percepción robótica se ocupa de estimar cantidades relevantes para la decisión a partir de datos de sensores. Para hacerlo, necesitamos una representación interna y un método para actualizar esta representación interna a lo largo del tiempo.
- Los algoritmos de filtrado probabilístico, como los filtros de partículas y los filtros de Kalman, son útiles para la percepción de los robots. Estas técnicas mantienen el estado de creencia, una distribución posterior sobre las variables de estado.
- Para generar movimiento, utilizamos espacios de configuración, donde un punto especifica todo lo que necesitamos saber para ubicar cada punto del cuerpo del robot. Por ejemplo, para un brazo robótico con dos articulaciones, una configuración consta de dos ángulos de articulación.
- Normalmente desacoplamos el problema de generación de movimiento en planificación de movimiento, que se ocupa de producir un plan, y control de seguimiento de trayectoria, que se ocupa de producir una política para las entradas de control (comandos de actuador) que resulta en la ejecución del plan.
- La planificación del movimiento se puede resolver mediante la búsqueda de gráficos mediante descomposición celular; utilizar algoritmos de planificación de movimiento aleatorios, que muestran hitos en el espacio de configuración continua; o utilizar la optimización de trayectoria, que puede hacer que una trayectoria en línea recta evite la colisión de forma iterativa aprovechando un campo de distancia con signo.
- Una ruta encontrada mediante un algoritmo de búsqueda se puede ejecutar utilizando la ruta como trayectoria de referencia para un controlador PID, que corrige constantemente los errores entre el lugar donde está el robot y donde se supone que debe estar, o mediante un control de par calculado, que agrega un Término de avance que hace uso de la dinámica inversa para calcular aproximadamente qué par enviar para avanzar a lo largo de la trayectoria.
- El control óptimo une la planificación del movimiento y el seguimiento de la trayectoria calculando una trayectoria óptima directamente sobre las entradas de control. Esto es especialmente fácil cuando tenemos costos cuadráticos y dinámica lineal, lo que da como resultado un regulador cuadrático lineal (LQR). Los métodos populares hacen uso de esto linealizando la dinámica y calculando aproximaciones del costo de segundo orden (ILQR).
- La planificación bajo incertidumbre une la percepción y la acción mediante la replanificación en línea (como el control predictivo de modelos) y acciones de recopilación de información que ayudan a la percepción.
- El aprendizaje por refuerzo se aplica en robótica, con técnicas que intentan reducir el número requerido de interacciones con el mundo real. Estas técnicas tienden a explotar los modelos, ya sea estimándolos y utilizándolos para planificar, o formando políticas que sean sólidas con respecto a diferentes parámetros posibles del modelo.
- La interacción con los humanos requiere la capacidad de coordinar las acciones del robot con las de ellos, lo que puede formularse como un juego. Generalmente descomponemos la solución en predicción, en la que utilizamos las acciones en curso de la persona para estimar lo que hará en el futuro, y acción, en la que utilizamos las predicciones para calcular el movimiento óptimo para el robot.
- Ayudar a los humanos también requiere la capacidad de aprender o inferir lo que quieren. Los robots pueden abordar esto aprendiendo la función de costo deseada que deben optimizar a partir del aporte humano, como demostraciones, correcciones o instrucción en lenguaje natural. Alternativamente, los robots pueden imitar el comportamiento humano y utilizar el aprendizaje por refuerzo para ayudar a afrontar el desafío de la generalización a nuevos estados.

## 6.2 Actividades guiadas de la UD04

| Notebook                                    | Enlace                                                                                                 |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------|
| Introducción a OpenCV                       |  Open in Colab        |
| PNG "1.-line.png" (Para Ejemplo1)           |  PNG 1.line.png       |
| PNG "1.-line_left.png" (Para Ejemplo1)      |  PNG 1.line left.png  |
| PNG "1.-line_right.png" (Para Ejemplo1)     |  PNG 1.line right.png |
| "1.-motionvideo.mp4" (Para Ejemplo1)        |                                                                                                        |
| Vehiculos de Braitenberg                    |  Open in Colab        |
| Ejemplos de Robots (AITK){:target="_blank"} |  Open in Colab        |

⌚15 de julio de 2025

## 6.3 Actividades entregables de la UD04

| Notebooks a entregar:                                                   | Enlace                                                                                                                                 |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| EX1 Ejercicios OpenCV                                                   |  <a href="#">Open in Colab</a>                      |
| PNG "EX1.-camp.png" (Para EX1)                                          |  <a href="#">PNG</a> <a href="#">EX1.camp.png</a>    |
| "EX1.-vtest.mp4" (Para EX1)                                             |                                                                                                                                        |
| EX2 Navegar con camara                                                  |  <a href="#">Open in Colab</a>                      |
| PNG "EX2_pista_1.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 1.png</a> |
| PNG "EX2_pista_2.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 2.png</a> |
| PNG "EX2_pista_3.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 3.png</a> |
| PNG "EX2_pista_4.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 4.png</a> |
| PNG "EX2_pista_5.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 5.png</a> |
| PNG "EX2_pista_6.png" (Para EX2 y 3)                                    |  <a href="#">PNG</a> <a href="#">EX2 pista 6.png</a> |
| EX3 Navegar con camara difusa                                           |  <a href="#">Open in Colab</a>                    |
| EX4 Generar datos de entrenamiento con camara                           |  <a href="#">Open in Colab</a>                    |
| EX5 Controlar robot con una Red Neuronal entrenada con los datos de EX4 |  <a href="#">Open in Colab</a>                    |
| EX6 Controlar robot con aprendizaje por refuerzo (NEAT)                 |  <a href="#">Open in Colab</a>                    |

⌚15 de julio de 2025

## 7. UD05

---

### 7.1 Aplicación de principios legales y éticos de la IA

---

#### Introducción

Al igual que en muchas otras disciplinas científicas y técnicas, a la hora de llevar acabo desarrollos en el campo de la inteligencia artificial, todo profesional debe ser capaz de diferenciar entre lo que sería posible desarrollar en este campo desde el punto de vista técnico y lo que es ético y legal.

El Libro Blanco sobre la inteligencia artificial, desarrollado por la Comisión Europea, pone de manifiesto cómo el desarrollo de la inteligencia artificial ha cambiado nuestra vida en los últimos años. Gracias a la rápida evolución de estas tecnologías, se están consiguiendo efectos muy beneficiosos para la sociedad como, por ejemplo, se ha mejorado la atención sanitaria haciendo más precisos los diagnósticos, se han producido mejoras en la agricultura, en los sistemas de producción, en la seguridad, etc.

Al mismo tiempo, debe tenerse en cuenta que la inteligencia artificial conlleva una serie de riesgos potenciales. Entre los posibles riesgos de la inteligencia artificial, el ya citado Libro Blanco menciona, a modo de ejemplo, la opacidad en la toma de todo tipo de decisiones, la posible discriminación de género, el uso de los modelos de inteligencia artificial y de los conjuntos de datos para fines delictivos, así como la pérdida de parte de nuestra privacidad.

En abril de 2018 se presentó la Estrategia Europea para la Inteligencia Artificial. Dicha estrategia propone un enfoque para el desarrollo de la inteligencia artificial en Europa, haciendo también énfasis en la regulación de su desarrollo y uso. Con esta estrategia la Comisión Europea se compromete a facilitar el avance científico, preservar el liderazgo tecnológico de la Unión Europea y garantizar que las nuevas tecnologías estén al servicio de los ciudadanos de manera que mejoren sus vidas respetando sus derechos.

Así, la intención de la Comisión Europea a lo largo de la década de 2020 es doble; por una parte, incentivar y financiar el desarrollo de la inteligencia artificial y, por otra, regular el uso que se haga de la misma.

#### Deontología profesional en inteligencia artificial

---

Se entiende por deontología la parte de la ética que trata de los deberes, especialmente de los que rigen una actividad profesional. Así, la deontología es el conjunto de deberes relacionados con el ejercicio de una determinada profesión.

Todo profesional dentro de su ámbito de trabajo se enfrenta a dilemas morales y éticos. Existe una serie de problemas éticos que se han presentado en el campo de la inteligencia artificial desde el comienzo de su desarrollo como disciplina científica. Algunos de los más comunes son los que se relacionan a continuación:

- Los procesos de automatización inteligente pueden causar la pérdida de puestos de trabajo, sobre todo los del personal con menor cualificación. De una u otra forma, este problema ético ha estado presente al menos desde la creación de la máquina de vapor en el siglo xviii y el comienzo de la revolución industrial. En la actualidad, todas las industrias son fundamentalmente dependientes del uso de ordenadores y, en algunos casos, también del empleo de algoritmos de inteligencia artificial. Como ejemplos de esta dependencia se pueden citar los programas de credit scoring, que son quienes de forma automática deciden a qué personas se les puede conceder un crédito bancario. Este trabajo, hasta la creación de estos sistemas, era realizado por humanos. Otro ejemplo podría ser cómo, en la actualidad, el uso de la robótica permite que ciertas operaciones rutinarias que se realizan en los almacenes sean llevadas a cabo por robots. Pero, si bien el uso de aplicaciones de inteligencia artificial puede conducir a la destrucción de algunos empleos, no es menos cierto que sirve para la generación de otros trabajos, en general más cualificados y, por tanto, con mayor remuneración.
- El que la inteligencia artificial reemplace a los humanos en muchas de sus tareas podría hacer que el ser humano quedase relegado, perdiendo toda utilidad para el trabajo. Aunque esta idea se puso de manifiesto hace ya muchas décadas por parte de escritores de ciencia-ficción como Alvin Toffler o Arthur C. Clarke, parece complicado que se llegue a convertir en una realidad.
- Los sistemas de inteligencia artificial se podrían emplear para usos ilegales o perjudiciales. Efectivamente, toda tecnología puede ser usada para fines ilegales, inmorales o destructivos y evitar que esto ocurra resulta prácticamente imposible.
- La utilización de sistemas de inteligencia artificial puede conducir a una pérdida de la responsabilidad individual. Un ejemplo de ello podría ser un médico que tome la decisión de aplicar un tratamiento a un paciente a partir de un diagnóstico equivocado proporcionado por un sistema de inteligencia artificial. En este caso, se produce un dilema relativo a si el médico, profesional formado, es el culpable por haber aceptado ese diagnóstico como bueno o si bien es el programador del sistema el único responsable de ese error. Además, si se dispone de sistemas expertos que son capaces de diagnosticar mejor que un médico, este podría estar siendo negligente en caso de no usar este tipo de sistemas para llevar a cabo sus diagnósticos.

- El éxito de la inteligencia artificial podría suponer el fin de la raza humana. En el caso de la inteligencia artificial, se trata de una tecnología que, por su propia definición, posiblemente llegase a tomar conciencia de sí misma y llevar a cabo sus propias decisiones. Así por ejemplo, en muchos libros de ciencia ficción se habla acerca de robots asesinos que tratan de acabar con la raza humana. Sin llegar a este extremo, por ejemplo, un error de estimación de un coche autónomo podría causar un accidente de graves consecuencias para los ocupantes de un vehículo. Pero, a favor del vehículo autónomo, se puede decir que ese tipo de errores los cometen los humanos con mucha mayor frecuencia.

En la actualidad, Europa produce más de un cuarto de todos los robots de servicios industriales y profesionales que se fabrican en el mundo y desempeña un papel importante en el desarrollo y uso de las aplicaciones informáticas para empresas y organizaciones, así como de las aplicaciones para el fomento de la administración digital y las aplicaciones de «empresas inteligentes».

Dentro del ecosistema de excelencia en inteligencia artificial que se pretende desarrollar en Europa, resulta necesario disponer de profesionales cualificados en este campo. Esto supone el desarrollo de las habilidades necesarias para trabajar en este campo y mejorar las cualificaciones profesionales de los trabajadores para adaptarlas a la transformación que implica esta tecnología. Dentro del currículo de los profesionales de este campo, el Libro Blanco sugiere la inclusión de competencias éticas.

Con el fin de lograr este propósito, en el año 2019 la Unión Europea publicó el documento titulado Directrices éticas para una IA fiable. El objetivo de estas directrices es promover una inteligencia artificial fiable. Según dicho documento, la fiabilidad de la inteligencia artificial se apoya en tres componentes que han de satisfacerse a lo largo de todo el ciclo de vida del sistema. En primer lugar, la inteligencia artificial debe ser lícita, es decir, cumplir todas las leyes y reglamentos aplicables; además, ha de ser ética, de modo que se garantice el respeto de los principios y valores éticos y, finalmente, tiene que ser robusta, tanto desde el punto de vista técnico como social, puesto que los sistemas de inteligencia artificial, incluso si las intenciones son buenas, pueden provocar daños a terceros.

## La privacidad de los datos

Si bien la mayor parte del desarrollo legislativo y la regulación de la protección de los datos de carácter personal, tanto en España como en Europa, se produjo a partir de 2010, ya la Constitución Española, promulgada en 1978, recoge en su artículo 18.4 que «La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos».

Es bien sabido que Internet ofrece grandes oportunidades a la ciudadanía, pero también supone un riesgo considerable en todo lo referente a su privacidad. Resulta por tanto necesaria la existencia de un marco normativo que haga efectivos los derechos de la ciudadanía en Internet, promoviendo la igualdad.

En la actualidad, en el Reino de España, la concreción y desarrollo del derecho fundamental de protección de las personas físicas en relación con el tratamiento de los datos personales; se encuentra recogido en la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y Garantía de los Derechos Digitales. Además, dentro del ámbito de la Unión Europea, en el año 2016 se adoptó el Reglamento (UE) del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de sus datos personales y a la libre circulación de esos datos.

La ya mencionada [Ley Orgánica 3/2018](#) recoge aspectos de gran interés e importancia para la ciudadanía. Se resumen a continuación algunos de los considerados como más relevantes:

- **Derecho de acceso:** todo ciudadano tiene derecho a solicitar acceso a los datos que de él dispone cualquier persona física o jurídica. Esta solicitud se canalizará a través del correspondiente responsable de los datos. Cuando el responsable disponga de una gran cantidad de datos relativos al afectado y este ejercite su derecho de acceso, deberá especificar si el acceso se refiere a la totalidad o a parte de esa información.
- **Derecho de acceso a los datos de personas fallecidas:** los familiares de una persona fallecida pueden dirigirse al responsable o encargado del tratamiento de los datos de una compañía que posea información relativa a su familiar, al objeto de solicitar el acceso a los datos del fallecido y permitir su rectificación o supresión, salvo que el fallecido hubiese designado expresamente un responsable de sus datos o bien hubiera prohibido expresamente el acceso a los mismos tras su fallecimiento.
- **Derecho de rectificación, supresión y limitación del tratamiento:** toda persona física tiene derecho a la rectificación, supresión o limitación del tratamiento que se haga de sus datos personales por parte de cualquier persona física o jurídica.
- **Sistemas de información crediticia:** en el caso de los sistemas de información crediticia, salvo prueba en contra, se presumirá lícito el tratamiento de datos relativo al incumplimiento de obligaciones dinerarias, financieras o de crédito por sistemas comunes de información crediticia, cuando los datos hayan sido facilitados por el acreedor o su representante, los datos se refieran a deudas vencidas que no hayan sido objeto de reclamación administrativa o judicial por parte del deudor y siempre que el acreedor haya informado al afectado en el contrato o en el momento de requerir el pago acerca de la posibilidad de inclusión en dichos sistemas. Finalmente, cabe destacar que los datos se mantendrán en el sistema únicamente mientras persista el incumplimiento y con un límite máximo de 5 años desde la fecha de vencimiento de la obligación dineraria, financiera o de crédito.

- **Tratamiento de la información con fines de videovigilancia:** en lo relativo a la videovigilancia, solamente se permite la captación de imágenes en la vía pública en la medida en que esta resulte imprescindible para garantizar la seguridad de las personas y de los bienes, teniendo los datos que ser suprimidos en el plazo máximo de 1 mes desde su captación, salvo que exista causa que justifique la necesidad de conservarlos para acreditar la comisión de actos que atenten contra la integridad de bienes o personas, en cuyo caso se tendrán que poner a disposición judicial en menos de 72 horas.
- **Sistemas de exclusión publicitaria:** cuando un ciudadano se pone en contacto con una compañía para pedir su exclusión publicitaria, esta le habrá de informar de los sistemas disponibles para dicha exclusión. Se puede restringir la recepción de publicidad no deseada mediante la inscripción gratuita y voluntaria en un fichero de exclusión publicitaria. Actualmente solo existe el fichero denominado Lista Robinson que está gestionado por la Asociación Española de Economía Digital. Esta lista debe ser consultada por quienes vayan a realizar una campaña publicitaria para excluir de la misma a las personas inscritas. Aquel ciudadano que esté inscrito en un fichero de exclusión publicitaria, únicamente recibirá publicidad de las compañías de las que sea cliente o de aquellas a las que haya autorizado expresamente a enviarle publicidad.
- **Derecho a la neutralidad de Internet y de acceso universal:** este derecho significa que los proveedores de servicios de Internet proporcionarán una oferta transparente de servicios sin discriminación por motivos técnicos o económicos. Además, se garantiza el acceso a Internet, independientemente de la condición personal, social, económica o geográfica de cada individuo.
- **Protección de los menores en Internet:** dados los posibles peligros a los que se exponen los menores en sus comunicaciones a través de Internet, la Ley Orgánica 3/2018 especifica que deben ser los tutores de los menores los que procuren que estos hagan un uso responsable de la red.
- **Derecho a la desconexión digital e intimidad frente al uso de dispositivos de videovigilancia, grabación y geolocalización:** la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y Garantía de los Derechos Digitales proporciona un marco regulatorio para los derechos de los trabajadores, alguno de ellos tan recientes como el derecho a la desconexión digital. El derecho a la desconexión digital consiste en que tanto los trabajadores como los empleados públicos tienen derecho a la desconexión de sus dispositivos electrónicos de trabajo a fin de garantizar, fuera de su jornada laboral, el respeto de su descanso e intimidad personal y familiar. Dentro del derecho a la desconexión digital se indica expresamente que este derecho se preservará también en el supuesto de realización total o parcial del trabajo desde el domicilio del empleado. También se regula en qué lugar y bajo qué condiciones resulta posible la instalación de dispositivos para la grabación de imágenes y sonido. Así, según se indica en la Ley, en ningún caso se admitirá la instalación de sistemas de grabación de sonidos ni de videovigilancia en lugares destinados al descanso o esparcimiento de los trabajadores tales como vestuarios, aseos, comedores, etc. En lo referente a la instalación de dispositivos geolocalizadores, los empleadores habrán de informar con carácter previo de la existencia de dichos dispositivos a sus empleados aunque podrán emplear la información obtenida de los mismos para el control de las labores del empleado aunque siempre dentro del marco legal.
- **El derecho al olvido:** consiste en que toda persona tiene derecho a que los motores de búsqueda en Internet eliminan de las listas de resultados que se obtuvieran los enlaces relativos a una persona cuando estos fuesen inadecuados, inexistentes, no pertinentes, etc. De igual manera también existe el derecho al olvido en los servicios de las redes sociales y equivalentes, teniendo el usuario derecho a que sean suprimidos de toda red sus datos personales cuando así lo solicite.

## Protección frente a errores

Los sistemas de inteligencia artificial cometén errores. Muchos de estos fallos son fácilmente detectables durante el desarrollo de los sistemas, pero otros pasan inadvertidos y se manifiestan durante el uso del mismo. Este tipo de errores pueden ser de muy diversa índole.

Así, por ejemplo, hoy en día existen sistemas de reconocimiento de imágenes muy sofisticados que son, capaces de etiquetar objetos. Gracias a la existencia de estas tecnologías se han conseguido desarrollos como el coche autónomo, pero dichos sistemas no están exentos de fallos. Seguidamente se exponen algunos errores relacionados con aplicaciones de la inteligencia artificial en distintos campos.

En 2015, el ingeniero informático Jacky Alciné se percató de que el algoritmo de reconocimiento de imágenes de Google Photos había etiquetado a algunos de sus amigos de raza negra como gorilas. Se trataba de un error producido por el algoritmo de inteligencia artificial de dicha aplicación e inmediatamente, Google pidió disculpas por ello prometiendo resolver el problema. Sin embargo, 3 años después, un reportaje publicado en la revista Wired demostró que este problema no estaba todavía resuelto. Realmente, la solución por la que había optado Google con el fin de evitar problemas con sus usuarios fue eliminar la etiqueta gorila así como la de algunos otros primates como chimpancé o mono. Es decir, las funcionalidades de la aplicación se habían limitado dado que no habían sido capaces de corregir el fallo. Este hecho pone de manifiesto la existencia de ciertas dificultades técnicas en el tratamiento de imágenes que todavía no han sido superadas.

El 18 de marzo de 2018 tuvo lugar el primer accidente con víctimas mortales causado por un coche autónomo. Este accidente se produjo cuando un coche autónomo de Uber arrolló en Tempe (Arizona, Estados Unidos) a una mujer que atravesaba una carretera de cuatro carriles empujando una bicicleta. El vehículo de Uber que atropelló a esta mujer operaba en modo autónomo pero asistido por un conductor quien, en caso de que existieran posibilidades de sufrir un accidente, debería de asumir los mandos del vehículo. Tras la investigación del suceso, se encontraron distintas causas que podrían haber influido en el mismo, algunas de las cuales se relacionan a continuación:

- En el momento del accidente, el coche se desplazaba a una velocidad de 69 km/h.
- El sistema podría haber visto al peatón 6 segundos antes de la colisión, pero durante 4,7 segundos no hizo nada por evitarla.
- El conductor que iba en el vehículo para garantizar la seguridad del mismo, dado que se trataba de un vehículo en pruebas, estaba distraído en el momento en el que produjo el accidente.
- La forma en la que el peatón cruzaba la carretera, empujando una bicicleta y en una hora de baja visibilidad, no era segura.
- La zona de la carretera por donde el peatón estaba cruzando tenía una señal que indicaba que estaba prohibido el paso a los peatones.
- Un peritaje realizado por expertos mostró que, en un primer momento, el software del coche autónomo falló en la detección del peatón. En este contexto, el que el peatón estuviese empujando una bicicleta (objeto metálico) pudo haber influido en la deficiente detección que el sistema hizo del mismo.

Tras el accidente, Uber suspendió las pruebas de su coche autónomo. Transcurridos unos meses, Uber emitió un informe en el que se afirmaba que con el grado de desarrollo que en esos momentos tenían sus coches autónomos, estos eran más seguros que los conducidos por humanos. El 20 de diciembre de 2018 Uber reanudó sus pruebas.

De entre de todos los posibles fallos que hoy en día presentan los asistentes de voz como Siri, Alexa o Cortana, se expone a continuación el que se produjo en una presentación que realizó el consejero delegado de Microsoft Satya Nadella en 2015. Se trataba de una conferencia de la empresa Salesforce.com, compañía dedicada al desarrollo de aplicaciones de software para la relación con el cliente.

En dicha conferencia Nadella trataba de mostrar las capacidades de análisis y entendimiento del habla de Cortana, asistente virtual de Microsoft, para lo que le preguntó al sistema de viva voz en inglés «Show me my most at-risk opportunities» lo que se podría traducir como «muéstrame mis oportunidades de mayor riesgo», lo que Cortana interpretó como «Show me to buy milk at this opportunity» que quiere decir algo así como «enséñame a comprar leche en esta oportunidad».

Aunque Nadella se lo repitió varias veces, Cortana no fue capaz de entenderlo y hubo de reformular la frase. La causa raíz de este fallo es que el sistema de reconocimiento de voz de Cortana, entrenado para entender a una persona hablando en inglés con acento norteamericano, no fue capaz de entender con la misma precisión lo que decía Nadella, nacido en la India aunque residente en Estados Unidos desde finales de la década de 1980.

En ingeniería de software, cuando se desarrolla un programa que no hace uso de inteligencia artificial, es el programador quien escribe el código fuente que interacciona con el usuario para llevar a cabo cierta tarea. En este contexto, las pruebas de software ayudan a garantizar que el programa funcione tal y como se espera. Sin embargo, en los sistemas de aprendizaje automático, la programación consiste en proporcionar al sistema ejemplos de cómo debe comportarse y hacer que este aprenda. Por tanto, es necesario llevar a cabo un minucioso proceso de prueba que garantice que con la información proporcionada el sistema ha sido capaz de aprender correctamente.

## Principios éticos

---

Dado que la IA es una tecnología poderosa, tenemos la obligación moral de utilizarla bien, promover los aspectos positivos y evitar o mitigar los negativos.

Los aspectos positivos son muchos. Por ejemplo, la IA puede salvar vidas mediante mejores diagnósticos médicos, nuevos descubrimientos médicos, una mejor predicción de fenómenos meteorológicos extremos y una conducción más segura con asistencia al conductor y (eventualmente) tecnologías de conducción autónoma. También hay muchas oportunidades para mejorar vidas. El programa AI for Humanitarian Action de Microsoft aplica la IA para recuperarse de desastres naturales, abordar las necesidades de los niños, proteger a los refugiados y promover los derechos humanos. El programa AI for Social Good de Google apoya el trabajo sobre la protección de la selva tropical, la jurisprudencia de derechos humanos, el monitoreo de la contaminación, la medición de las emisiones de combustibles fósiles, el asesoramiento en crisis, la verificación de noticias, la prevención del suicidio, el reciclaje y otros temas. El Centro de Ciencia de Datos para el Bien Social de la Universidad de Chicago aplica el aprendizaje automático a problemas de justicia penal, desarrollo económico, educación, salud pública, energía y medio ambiente.

Las aplicaciones de IA en el manejo de cultivos y la producción de alimentos ayudan a alimentar al mundo. La optimización de los procesos empresariales mediante el aprendizaje automático hará que las empresas sean más productivas, aumentará la riqueza y generará más empleo. La automatización puede reemplazar las tareas tediosas y peligrosas que enfrentan muchos trabajadores y liberarlos para concentrarse en aspectos más interesantes. Las personas con discapacidad se beneficiarán de la asistencia basada en inteligencia artificial para ver, oír y moverse. La traducción automática ya permite comunicarse entre personas de diferentes culturas. Las soluciones de IA basadas en software tienen un costo marginal de producción casi nulo y, por lo tanto, tienen el potencial de democratizar el acceso a la tecnología avanzada (incluso cuando otros aspectos del software tienen el potencial de centralizar el poder).

A pesar de estos muchos aspectos positivos, no debemos ignorar los negativos. Muchas tecnologías nuevas han tenido efectos secundarios negativos no deseados: la fisión nuclear provocó Chernobyl y la amenaza de destrucción global; el motor de

combustión interna trajo contaminación del aire, calentamiento global y la pavimentación del paraíso. Otras tecnologías pueden tener efectos negativos incluso cuando se utilizan según lo previsto, como el gas sarín, los rifles AR-15 y las solicitudes telefónicas. La automatización creará riqueza, pero en las condiciones económicas actuales gran parte de esa riqueza fluirá hacia los propietarios de los sistemas automatizados, lo que conducirá a una mayor desigualdad de ingresos. Esto puede ser perjudicial para una sociedad que funcione bien. En los países en desarrollo, el camino tradicional hacia el crecimiento a través de la fabricación de bajo costo para la exportación puede verse cortado, a medida que los países ricos adopten instalaciones de fabricación en el país totalmente automatizadas. Nuestras decisiones éticas y de gobernanza dictarán el nivel de desigualdad que generará la IA.

Todos los científicos e ingenieros enfrentan consideraciones éticas sobre qué proyectos deben o no emprender y cómo pueden asegurarse de que la ejecución del proyecto sea segura y beneficiosa. En 2010, el Consejo de Investigación en Ingeniería y Ciencias Físicas del Reino Unido celebró una reunión para desarrollar un conjunto de Principios de la Robótica. En los años siguientes, otras agencias gubernamentales, organizaciones sin fines de lucro y empresas crearon conjuntos de principios similares. La esencia es que cada organización que crea tecnología de IA, y todos los miembros de la organización, tienen la responsabilidad de asegurarse de que la tecnología contribuya al bien y no al daño. Los principios más comúnmente citados son:

- Garantizar la seguridad
- Establecer responsabilidad
- Garantizar la equidad
- Defender los derechos y valores humanos.
- Respetar la privacidad
- Reflejar diversidad/inclusión
- Promover la colaboración
- Evitar la concentración de poder.
- Proporcionar transparencia.
- Reconocer las implicaciones legales/políticas.
- Limitar los usos nocivos de la IA.
- Contemplar las implicaciones para el empleo.

Tenga en cuenta que muchos de los principios, como "garantizar la seguridad", son aplicables a todos los sistemas de software o hardware, no solo a los sistemas de inteligencia artificial. Varios principios están redactados de manera vaga, lo que dificulta su medición o aplicación. Esto se debe en parte a que la IA es un gran campo con muchos subcampos, cada uno de los cuales tiene un conjunto diferente de normas históricas y diferentes relaciones entre los desarrolladores de IA y las partes interesadas. Mittelstadt (2019) sugiere que cada uno de los subcampos debería desarrollar pautas procesables y precedentes de casos más específicos.

#### **Armas autónomas letales**

La ONU define un arma letal autónoma como aquella que localiza, selecciona y ataca (es decir, mata) objetivos humanos sin supervisión humana. Varias armas cumplen algunos de estos criterios. Por ejemplo, las minas terrestres se utilizan desde el siglo XVII: pueden seleccionar y atacar objetivos de forma limitada según el grado de presión ejercida o la cantidad de metal presente, pero no pueden salir y localizar objetivos por sí mismas. (Las minas terrestres están prohibidas en virtud del Tratado de Ottawa). Los misiles guiados, utilizados desde la década de 1940, pueden perseguir objetivos, pero un ser humano debe apuntarlos en la dirección general correcta. Los cañones de disparo automático controlados por radar se han utilizado para defender buques de guerra desde la década de 1970; Su objetivo principal es destruir los misiles entrantes, pero también podrían atacar aviones tripulados. Aunque la palabra "autónoma" se utiliza a menudo para describir vehículos aéreos no tripulados o drones, la mayoría de estas armas son pilotadas de forma remota y requieren la actuación humana de la carga letal.

En el momento de escribir este artículo, varios sistemas de armas parecen haber cruzado la línea hacia la plena autonomía. Por ejemplo, el misil Harop de Israel es una "munición merodeadora" con una envergadura de tres metros y una ojiva de cincuenta libras. Busca durante hasta seis horas en una región geográfica determinada cualquier objetivo que cumpla un criterio determinado y luego lo destruye. El criterio podría ser "emite una señal de radar que se asemeja a un radar antiaéreo" o "parece un tanque". El fabricante turco STM anuncia su cuadricóptero Kargu, que transporta hasta 1,5 kg de explosivos, como capaz de "impactar de forma autónoma... objetivos seleccionados en imágenes... rastrear objetivos en movimiento... antipersonal... reconocimiento facial".

Las armas autónomas han sido llamadas la "tercera revolución en la guerra" después de la pólvora y las armas nucleares. Su potencial militar es obvio. Por ejemplo, pocos expertos dudan de que los aviones de combate autónomos derrotarían a cualquier piloto humano. Los aviones, tanques y submarinos autónomos pueden ser más baratos, más rápidos, más maniobrables y tener mayor alcance que sus homólogos tripulados.

Desde 2014, las Naciones Unidas en Ginebra han llevado a cabo debates periódicos bajo los auspicios de la Convención sobre Ciertas Armas Convencionales (CAC) sobre la cuestión de si se deben prohibir las armas letales autónomas. En el momento de

redactar este informe, 30 naciones, cuyo tamaño varía desde China hasta la Santa Sede, han declarado su apoyo a un tratado internacional, mientras que otros países clave, incluidos Israel, Rusia, Corea del Sur y Estados Unidos, se oponen a un tratado de prohibición.

El debate sobre las armas autónomas incluye aspectos legales, éticos y prácticos. Las cuestiones jurídicas se rigen principalmente por la CCW, que exige la posibilidad de discriminar entre combatientes y no combatientes, el juicio sobre la necesidad militar de un ataque y la evaluación de la proporcionalidad entre el valor militar de un objetivo y la posibilidad de daños colaterales.

La viabilidad de cumplir estos criterios es una cuestión de ingeniería, cuya respuesta sin duda cambiará con el tiempo. En la actualidad, la discriminación parece factible en algunas circunstancias y sin duda mejorará rápidamente, pero la necesidad y la proporcionalidad no son factibles actualmente: requieren que las máquinas hagan juicios subjetivos y situacionales que son considerablemente más difíciles que las tareas relativamente simples de buscar y atacar objetivos potenciales. . Por estas razones, sería legal usar armas autónomas solo en circunstancias en las que un operador humano pueda predecir razonablemente que la ejecución de la misión no resultará en que los civiles sean atacados o que las armas realicen ataques innecesarios o desproporcionados. Esto significa que, por el momento, sólo se pueden realizar misiones muy restringidas con armas autónomas.

Desde el punto de vista ético, algunos consideran simplemente moralmente inaceptable delegar la decisión de matar humanos a una máquina. Por ejemplo, el embajador de Alemania en Ginebra ha declarado que "no aceptará que la decisión sobre la vida o la muerte sea tomada únicamente por un sistema autónomo", mientras que Japón "no tiene ningún plan para desarrollar robots con humanos fuera del circuito, que tal vez sean capaces de cometer asesinato". El general Paul Selva, en ese momento el segundo oficial militar de Estados Unidos, dijo en 2017: "No creo que sea razonable que pongamos a los robots a cargo de si matamos o no una vida humana". Finalmente, António Guterres, jefe de las Naciones Unidas, afirmó en 2019 que "las máquinas con el poder y la discreción de quitar vidas sin participación humana son políticamente inaceptables, moralmente repugnantes y deberían estar prohibidas por el derecho internacional".

Más de 140 ONG en más de 60 países forman parte de la Campaña para detener a los robots asesinos, y una carta abierta organizada en 2015 por el Future of Life Institute fue firmada por más de 4.000 investigadores de IA2 y 22.000 más.

En contra de esto, se puede argumentar que a medida que la tecnología mejore, debería ser posible desarrollar armas que tengan menos probabilidades que los soldados o pilotos humanos de causar víctimas civiles. (También existe el importante beneficio de que las armas autónomas reducen la necesidad de soldados humanos y pilotos corren el riesgo de morir.) Los sistemas autónomos no sucumbirán a la fatiga, la frustración, la histeria, el miedo, la ira o la venganza, y no necesitan "disparar primero, hacer preguntas después" (Arkin, 2015). Así como las municiones guiadas han reducido el daño colateral En comparación con las bombas no guiadas, se puede esperar que las armas inteligentes mejoren aún más la precisión de los ataques. (En contra de esto, ver Benjamin (2013) para un análisis de las víctimas de la guerra con aviones no tripulados). Esta, aparentemente, es la posición de Estados Unidos en la última ronda de negociaciones en Ginebra.

Quizás contrariamente a lo intuitivo, Estados Unidos es también una de las pocas naciones cuyas propias políticas actualmente excluyen el uso de armas autónomas. La hoja de ruta del Departamento de Defensa de Estados Unidos (DOD) de 2011 dice: "En el futuro previsible, las decisiones sobre el uso de la fuerza [por sistemas autónomos] y la elección de qué objetivos individuales atacar con fuerza letal se mantendrán bajo control humano". La razón principal de esta política es práctica: los sistemas autónomos no son lo suficientemente confiables como para confiarles decisiones militares.

La cuestión de la confiabilidad pasó a primer plano el 26 de septiembre de 1983, cuando la pantalla de la computadora del oficial de misiles soviético Stanislav Petrov mostró una alerta de un ataque con misiles inminente. Según el protocolo, Petrov debería haber iniciado un contraataque nuclear, pero sospechaba que la alerta era un error y lo trató como tal. Tenía razón y la Tercera Guerra Mundial se evitó (por poco). No sabemos qué habría pasado si no hubiera habido ningún ser humano en el circuito.

La confiabilidad es una preocupación muy seria para los comandantes militares, quienes conocen bien la complejidad de las situaciones en el campo de batalla. Los sistemas de aprendizaje automático que funcionan perfectamente durante la capacitación pueden tener un rendimiento deficiente cuando se implementan. Los ciberataques contra armas autónomas podrían provocar bajas por fuego amigo; desconectar el arma de toda comunicación puede evitarlo (suponiendo que aún no haya sido comprometida), pero entonces el arma no puede recuperarse si no funciona correctamente.

La cuestión práctica primordial con las armas autónomas es que son armas de destrucción masiva escalables, en el sentido de que la escala de un ataque que se puede lanzar es proporcional a la cantidad de hardware que uno puede permitirse desplegar. Un cuadricóptero de cinco centímetros de diámetro puede transportar una carga explosiva letal, y un millón de ellos caben en un contenedor de transporte normal. Precisamente porque son autónomas, estas armas no necesitarían un millón de supervisores humanos para hacer su trabajo.

Como armas de destrucción masiva, las armas autónomas tienen ventajas para el atacante en comparación con las armas nucleares y los bombardeos masivos: dejan la propiedad intacta y pueden usarse selectivamente para eliminar sólo a aquellos que puedan amenazar a una fuerza ocupante. Sin duda, podrían utilizarse para eliminar a todo un grupo étnico o a todos los seguidores de una religión determinada. En muchas situaciones, también serían imposibles de rastrear. Estas características los hacen particularmente atractivos para los actores no estatales.

Estas consideraciones –particularmente aquellas características que benefician al atacante– sugieren que las armas autónomas reducirán la seguridad global y nacional de todas las partes. La respuesta racional de los gobiernos parece ser involucrarse en discusiones sobre control de armas en lugar de una carrera armamentista.

Sin embargo, el proceso de diseño de un tratado no está exento de dificultades. La IA es una tecnología de doble uso: las tecnologías de IA que tienen aplicaciones pacíficas, como control de vuelo, seguimiento visual, cartografía, navegación y planificación multiagente, pueden aplicarse fácilmente con fines militares. Es fácil convertir un cuadricóptero autónomo en un arma simplemente colocando un explosivo y ordenándole que busque un objetivo. Para abordar esto será necesario implementar cuidadosamente regímenes de cumplimiento con la cooperación de la industria, como ya ha demostrado con cierto éxito la Convención sobre Armas Químicas.

### Vigilancia, seguridad y privacidad

En 1976, Joseph Weizenbaum advirtió que la tecnología de reconocimiento automatizado de voz podría dar lugar a escuchas telefónicas generalizadas y, por tanto, a una pérdida de libertades civiles. Hoy en día, esa amenaza se ha hecho realidad: la mayoría de las comunicaciones electrónicas pasan por servidores centrales que pueden ser monitoreados y las ciudades están repletas de micrófonos y cámaras que pueden identificar y rastrear a las personas basándose en su voz, rostro y modo de andar. La vigilancia que solía requerir recursos humanos costosos y escasos ahora puede realizarse a gran escala mediante máquinas.

En 2018, había hasta 350 millones de cámaras de vigilancia en China y 70 millones en Estados Unidos. China y otros países han comenzado a exportar tecnología de vigilancia a países de baja tecnología, algunos con reputación de maltratar a sus ciudadanos y apuntar desproporcionadamente a comunidades marginadas. Los ingenieros de IA deben tener claro qué usos de la vigilancia son compatibles con los derechos humanos y negarse a trabajar en aplicaciones que sean incompatibles.

A medida que más instituciones nuestras operan en línea, nos volvemos más vulnerables al cibercrimen (phishing, fraude con tarjetas de crédito, redes de bots, ransomware) y al ciberterrorismo (incluidos ataques potencialmente mortales como el cierre de hospitales y plantas de energía o el control de vehículos autónomos). El aprendizaje automático puede ser una herramienta poderosa para ambas partes en la batalla de la ciberseguridad. Los atacantes pueden utilizar la automatización para detectar inseguridades y pueden aplicar aprendizaje reforzado para intentos de phishing y chantaje automatizado. Los defensores pueden utilizar el aprendizaje no supervisado para detectar patrones de tráfico entrante anómalos (Chandola et al., 2009; Malhotra et al., 2015) y diversas técnicas de aprendizaje automático para detectar fraude (Fawcett y Provost, 1997; Bolton y Hand, 2002). A medida que los ataques se vuelven más sofisticados, todos los ingenieros, no solo los expertos en seguridad, tienen una mayor responsabilidad de diseñar sistemas seguros desde el principio. Un pronóstico (Kanal, 2017) sitúa el mercado del aprendizaje automático en ciberseguridad en aproximadamente 100 mil millones de dólares para 2021..

A medida que interactuamos con las computadoras durante una cantidad cada vez mayor de nuestra vida diaria, los gobiernos y las corporaciones recopilan más datos sobre nosotros. Los recolectores de datos tienen la responsabilidad moral y legal de ser buenos administradores de los datos que poseen. En los EE. UU., la Ley de Responsabilidad y Portabilidad del Seguro Médico (HIPAA) y la Ley de Privacidad y Derechos Educativos de la Familia (FERPA) protegen la privacidad de los registros médicos y de los estudiantes. El Reglamento General de Protección de Datos (GDPR) de la Unión Europea exige que las empresas diseñen sus sistemas teniendo en cuenta la protección de datos y exige que obtengan el consentimiento del usuario para cualquier recopilación o procesamiento de datos.

En contraposición al derecho del individuo a la privacidad está el valor que la sociedad obtiene al compartir datos. Queremos poder detener a los terroristas sin oprimir la disidencia pacífica y queremos curar enfermedades sin comprometer el derecho de ningún individuo a mantener en privado su historial médico. Una práctica clave es la desidentificación: eliminar la información de identificación personal (como el nombre y el número de seguro social) para que los investigadores médicos puedan utilizar los datos para promover el bien común. El problema es que los datos no identificados compartidos pueden estar sujetos a reidentificación. Por ejemplo, si los datos excluyen el nombre, el número de seguro social y la dirección postal, pero incluyen la fecha de nacimiento, el sexo y el código postal, entonces, como lo muestra Latanya Sweeney (2000), el 87% de la población estadounidense puede ser reidentificado de forma única. Sweeney enfatizó este punto al volver a identificar el historial médico del gobernador de su estado cuando ingresó en el hospital. En la competencia del Premio Netflix, se publicaron registros anónimos de calificaciones de películas individuales y se pidió a los competidores que idearan un algoritmo de aprendizaje automático que pudiera predecir con precisión qué películas le gustarían a un individuo. Pero los investigadores pudieron volver a identificar a usuarios individuales haciendo coincidir la fecha de una clasificación en la base de datos de Netflix con la fecha de una clasificación similar en Internet Movie Database (IMDB), donde los usuarios a veces usan sus nombres reales (Narayanan y Shmatikov, 2006).).

Este riesgo se puede mitigar en cierta medida generalizando campos: por ejemplo, reemplazando la fecha de nacimiento exacta solo con el año de nacimiento, o un rango más amplio como "20-30 años". Eliminar un campo por completo puede verse como una forma de generalizar a "cualquiera". Pero la generalización por sí sola no garantiza que los registros estén a salvo de una reidentificación; Es posible que solo haya una persona en el código postal 94720 que tenga entre 90 y 100 años. Una propiedad útil es el k-anonimato: una base de datos se k-anonimiza si cada registro en la base de datos es indistinguible de al menos k-1 otros registros. Si hay registros que son más únicos que este, habría que generalizarlos aún más.

Una alternativa a compartir registros no identificados es mantener todos los registros privados, pero permitir consultas agregadas. Se proporciona una API para consultas a la base de datos y las consultas válidas reciben una respuesta que resume los datos con un recuento o promedio. Pero no se da ninguna respuesta si ello violaría ciertas garantías de privacidad. Por ejemplo, podríamos permitir que un epidemiólogo preguntara, para cada código postal, el porcentaje de personas con cáncer. Para códigos postales con al menos  $n$  personas, se daría un porcentaje (con una pequeña cantidad de ruido aleatorio), pero no se daría respuesta para códigos postales con menos de  $n$  personas.

Se debe tener cuidado para protegerse contra la desidentificación mediante consultas múltiples. Por ejemplo, si la consulta "salario promedio y número de empleados de la empresa XYZ de 30 a 40 años" da la respuesta [81.234,12 €] y la consulta "salario promedio y número de empleados de la empresa XYZ de 30 a 41 años" da la respuesta [ 81.199,13€], y si usamos LinkedIn para encontrar al hombre de 41 años en la compañía XYZ, entonces lo hemos identificado con éxito y podemos calcular su salario exacto, a pesar de que todas las respuestas involucraron a 12 o más personas. El sistema debe diseñarse cuidadosamente para protegerse contra esto, con una combinación de límites en las consultas que se pueden realizar (tal vez sólo se pueda consultar un conjunto predefinido de rangos de edad que no se superpongan) y la precisión de los resultados (tal vez ambas consultas den la respuesta "alrededor de 81.000 €").

Una garantía más sólida es la privacidad diferencial, que garantiza que un atacante no pueda utilizar consultas para volver a identificar a ningún individuo en la base de datos, incluso si el atacante puede realizar múltiples consultas y tiene acceso a bases de datos vinculadas separadas. La respuesta a la consulta emplea un algoritmo aleatorio que agrega una pequeña cantidad de ruido al resultado. En otras palabras, el hecho de que una persona decida participar o no en la base de datos no supone una diferencia apreciable en las respuestas que cualquiera pueda obtener y, por lo tanto, no existe ningún desincentivo de privacidad para participar. Muchas bases de datos están diseñadas para garantizar una privacidad diferencial.

Hasta ahora hemos considerado la cuestión de compartir datos no identificados de una base de datos central. Un enfoque llamado aprendizaje federado no tiene una base de datos central; en cambio, los usuarios mantienen sus propias bases de datos locales que mantienen la privacidad de sus datos. Sin embargo, pueden compartir parámetros de un modelo de aprendizaje automático que se mejora con sus datos, sin el riesgo de revelar ninguno de los datos privados. Imagine una aplicación de comprensión del habla que los usuarios puedan ejecutar localmente en su teléfono. La aplicación contiene una red neuronal básica, que luego se mejora mediante entrenamiento local sobre las palabras que se escuchan en el teléfono del usuario. Periódicamente, los propietarios de la aplicación encuestan a un subconjunto de usuarios y les preguntan los valores de los parámetros de su sistema local mejorado. red, pero no para ninguno de sus datos sin procesar. Los valores de los parámetros se combinan para formar un nuevo modelo mejorado que luego se pone a disposición de todos los usuarios, para que todos obtengan el beneficio de la capacitación realizada por otros usuarios.

Para que este esquema preserve la privacidad, debemos poder garantizar que los parámetros del modelo compartidos por cada usuario no puedan someterse a ingeniería inversa. Si enviamos los parámetros sin procesar, existe la posibilidad de que un adversario que los inspeccione pueda deducir si, por ejemplo, una determinada palabra había sido escuchada por el teléfono del usuario. Una forma de eliminar este riesgo es mediante la agregación segura (Bonawitz et al., 2017). La idea es que el servidor central no necesite conocer el valor exacto del parámetro de cada usuario distribuido; sólo necesita saber el valor promedio de cada parámetro, entre todos los usuarios encuestados. De modo que cada usuario puede disfrazar los valores de sus parámetros agregando una máscara única a cada valor; Siempre que la suma de las máscaras sea cero, el servidor central podrá calcular el promedio correcto. Los detalles del protocolo garantizan que sea eficiente en términos de comunicación (menos de la mitad de los bits transmitidos corresponden a enmascaramiento), que sea robusto ante usuarios individuales que no responden y que sea seguro frente a usuarios adversarios, espías o incluso un servidor central adversario.

### Equidad y parcialidad

El aprendizaje automático está aumentando y, a veces, reemplazando la toma de decisiones humanas en situaciones importantes: qué préstamo se aprueba, en qué vecindarios se despliegan los agentes de policía, quién obtiene la libertad provisional o la libertad condicional. Pero los modelos de aprendizaje automático pueden perpetuar los prejuicios sociales. Consideremos el ejemplo de un algoritmo para predecir si es probable que los acusados reincidan y, por tanto, si deberían ser puestos en libertad antes del juicio. Bien podría ser que tal sistema recoja los prejuicios raciales o de género de los jueces humanos a partir de los ejemplos del conjunto de capacitación. Los diseñadores de sistemas de aprendizaje automático tienen la responsabilidad moral de garantizar que sus sistemas sean realmente justos. En ámbitos regulados como el crédito, la educación, el empleo y la vivienda, también tienen una responsabilidad legal. Pero ¿qué es la justicia? Hay múltiples criterios; Aquí hay seis de los conceptos más utilizados:

- **Justicia individual:** Requisito de que los individuos sean tratados de manera similar a otros individuos similares, independientemente de en qué clase se encuentren.
- **Equidad de grupo:** requisito de que dos clases sean tratadas de manera similar, según lo medido por alguna estadística resumida.
- **Equidad por desconocimiento:** si eliminamos los atributos de raza y género del conjunto de datos, entonces podría parecer que el sistema no puede discriminar esos atributos. Desafortunadamente, sabemos que los modelos de aprendizaje automático pueden predecir variables latentes (como la raza y el género). dadas otras variables correlacionadas (como el código postal y la ocupación). Además, eliminar esos atributos hace imposible verificar la igualdad de oportunidades o la igualdad de resultados.

Aún así, algunos países (por ejemplo, Alemania) han elegido este enfoque para sus estadísticas demográficas (independientemente de si se utilizan modelos de aprendizaje automático o no).

- **Igual resultado:** La idea de que cada clase demográfica obtiene los mismos resultados; Tienen paridad demográfica. Por ejemplo, supongamos que tenemos que decidir si debemos aprobar las solicitudes de préstamo; el objetivo es aprobar a aquellos solicitantes que pagarán el préstamo y no a aquellos que no cumplirán con el mismo. La paridad demográfica dice que tanto hombres como mujeres deberían tener el mismo porcentaje de préstamos aprobados. Tenga en cuenta que este es un criterio de equidad grupal que no garantiza la equidad individual; un solicitante bien calificado podría ser rechazado y un solicitante mal calificado podría ser aprobado, siempre y cuando los porcentajes generales sean iguales. Además, este enfoque favorece la corrección de sesgos pasados por encima de la precisión de la predicción. Si un hombre y una mujer son iguales en todos los sentidos, excepto que la mujer recibe un salario más bajo por el mismo trabajo, ¿debería aprobarse porque sería igual si no fuera por sesgos históricos, o debería denegarse porque el salario más bajo no influye en ¿Esto la hace más propensa a incumplir?
- **Igualdad de oportunidades:** La idea de que las personas que realmente tienen la capacidad de pagar el préstamo deben tener las mismas posibilidades de ser correctamente clasificadas como tales, independientemente de su sexo. Este enfoque también se llama "equilibrio". Puede conducir a resultados desiguales e ignora el efecto del sesgo en los procesos sociales que produjeron los datos de capacitación.
- **Igual impacto:** Personas con similar probabilidad de pagar el préstamo deberían tener la misma utilidad esperada, independientemente de la clase a la que pertenezcan. Esto va más allá de la igualdad de oportunidades en el sentido de que considera tanto los beneficios de una predicción verdadera como los costos de una predicción falsa.

Examinemos cómo se desarrollan estas cuestiones en un contexto particular. COMPAS es un sistema comercial de puntuación de reincidencia (reincidencia). Asigna a un acusado en un caso penal una puntuación de riesgo, que luego el juez utiliza para ayudar a tomar decisiones: ¿Es seguro liberar al acusado antes del juicio o debería encarcelarlo? En caso de ser declarado culpable, ¿cuánto debería durar la sentencia? ¿Debería concederse la libertad condicional? Dada la importancia de estas decisiones, el sistema ha sido objeto de un intenso escrutinio (Dressel y Farid, 2018).

COMPAS está diseñado para estar bien calibrado: todos los individuos a los que el algoritmo les da la misma puntuación deben tener aproximadamente la misma probabilidad de reincidir, independientemente de su raza. Por ejemplo, entre todas las personas a las que el modelo asigna una puntuación de riesgo de 7 sobre 10, el 60% de los blancos y el 61% de los negros reinciden. Por tanto, los diseñadores afirman que cumple con el objetivo de equidad deseado.

Por otro lado, COMPAS no logra la igualdad de oportunidades: la proporción de aquellos que no reincidieron pero fueron calificados falsamente como de alto riesgo fue del 45% para los negros y del 23% para los blancos. En el caso Estado contra Loomis, donde un juez se basó en COMPAS para determinar la sentencia del acusado, Loomis argumentó que el secreto funcionamiento interno del algoritmo violaba sus derechos al debido proceso. Aunque la Corte Suprema de Wisconsin determinó que la sentencia dictada no sería diferente sin COMPAS en este caso, sí emitió advertencias sobre la precisión del algoritmo y los riesgos para los acusados minoritarios. Otros investigadores han cuestionado si es apropiado utilizar algoritmos en aplicaciones como la sentencia.

Podríamos esperar un algoritmo que esté bien calibrado y ofrezca igualdad de oportunidades, pero, como Kleinberg et al. (2016), eso es imposible. Si las clases base son diferentes, entonces cualquier algoritmo que esté bien calibrado no necesariamente brindará igualdad de oportunidades, y viceversa. ¿Cómo podemos sopesar los dos criterios? El mismo impacto es una posibilidad. En el caso de COMPAS, esto significa sopesar la utilidad negativa de que los acusados sean clasificados falsamente como de alto riesgo y pierdan su libertad, versus el costo para la sociedad de cometer un delito adicional, y encontrar el punto que optimice la compensación. Esto es complicado porque hay múltiples costos a considerar. Hay costos individuales: un acusado que es encarcelado injustamente sufre una pérdida, al igual que la víctima de un acusado que fue liberado injustamente y reincide. Pero más allá de eso, hay costos grupales: todos tienen cierto temor de ser encarcelados injustamente o de ser víctimas de un delito, y todos los contribuyentes contribuyen a los costos de las cárceles y los tribunales. Si valoramos esos temores y costos en proporción al tamaño de un grupo, entonces la utilidad para la mayoría puede llegar a expensas de una minoría.

Otro problema con la idea de la puntuación de reincidencia, independientemente del modelo utilizado, es que no disponemos de datos reales imparciales. Los datos no nos dicen quién ha cometido un delito; todo lo que sabemos es quién ha sido condenado por un delito. Si los agentes, el juez o el jurado que lo arrestan están sesgados, entonces los datos estarán sesgados. Si más agentes patrullan algunos lugares, los datos estarán sesgados en contra de las personas en esos lugares. Sólo los acusados que son liberados son candidatos a volver a ser condenados, por lo que si los jueces que toman las decisiones de liberación están sesgados, los datos pueden estar sesgados. Si se supone que detrás del conjunto de datos sesgados hay un conjunto de datos subyacente, desconocido e imparcial que ha sido corrompido por un agente con sesgos, entonces existen técnicas para recuperar una aproximación a los datos imparciales. Jiang y Nachum (2019) describen varios escenarios y las técnicas involucradas.

Un riesgo más es que el aprendizaje automático pueda utilizarse para justificar sesgos. Si las decisiones las toma un humano sesgado después de consultar con un sistema de aprendizaje automático, el humano puede decir "así es como mi interpretación del modelo respalda mi decisión, por lo que no deberías cuestionar mi decisión". Pero otras interpretaciones podrían conducir a una decisión contraria.

A veces, la justicia significa que debemos reconsiderar la función objetivo, no los datos o el algoritmo. Por ejemplo, al tomar decisiones de contratación laboral, si el objetivo es contratar candidatos con las mejores calificaciones, corremos el riesgo de

recompensar injustamente a quienes han tenido oportunidades educativas ventajosas a lo largo de su vida, imponiendo así los límites de clase. Pero si el objetivo es contratar candidatos con la mejor capacidad para aprender en el trabajo, tenemos más posibilidades de trascender las fronteras de clase y elegir entre un grupo más amplio. Muchas empresas tienen programas diseñados para estos solicitantes y descubren que después de un año de capacitación, los empleados contratados de esta manera obtienen tan buenos resultados como los candidatos tradicionales. De manera similar, sólo el 18% de los graduados en ciencias de la computación en EE.UU. son mujeres, pero algunas escuelas, como la Universidad Harvey Mudd, han logrado una paridad del 50% con un enfoque que se centra en alentar y retener a quienes inician el programa de ciencias de la computación, especialmente aquellos que comienzan con menos experiencia en programación.

Una última complicación es decidir qué clases merecen protección. En Estados Unidos, la Ley de Vivienda Justa reconoció siete clases protegidas: raza, color, religión, origen nacional, sexo, discapacidad y situación familiar. Otras leyes locales, estatales y federales reconocen otras clases, incluida la orientación sexual y el embarazo, el estado civil y el estado de veterano. ¿Es justo que estas clases cuenten para algunas leyes y no para otras? El derecho internacional de los derechos humanos, que abarca un amplio conjunto de clases protegidas, es un marco potencial para armonizar las protecciones entre varios grupos.

Incluso en ausencia de sesgo social, la disparidad en el tamaño de la muestra puede generar resultados sesgados. En la mayoría de los conjuntos de datos habrá menos ejemplos de entrenamiento de individuos de clases minoritarias que de individuos de clases mayoritarias. Los algoritmos de aprendizaje automático brindan mayor precisión con más datos de entrenamiento, lo que significa que los miembros de clases minoritarias experimentarán una menor precisión. Por ejemplo, Buolamwini y Gebru (2018) examinaron un servicio de identificación de género por visión por computadora y descubrieron que tenía una precisión casi perfecta para los hombres de piel clara y una tasa de error del 33 % para las mujeres de piel oscura. Es posible que un modelo restringido no pueda ajustarse simultáneamente a la clase mayoritaria y minoritaria; un modelo de regresión lineal podría minimizar el error promedio ajustando solo la clase mayoritaria, y en un modelo SVM, todos los vectores de soporte podrían corresponder a miembros de la clase mayoritaria.

El sesgo también puede entrar en juego en el proceso de desarrollo de software (ya sea que el software implique aprendizaje automático o no). Es más probable que los ingenieros que depuran un sistema noten y solucionen los problemas que les son aplicables. Por ejemplo, es difícil darse cuenta de que el diseño de una interfaz de usuario no funcionará para personas daltónicas a menos que usted sea daltónico, o que una traducción al idioma urdu sea defectuosa si no habla urdu.

¿Cómo podemos defendernos de estos prejuicios? Primero, comprenda los límites de los datos que está utilizando. Se ha sugerido que los conjuntos de datos (Gebru et al., 2018; Hind et al., 2018) y los modelos (Mitchell et al., 2019) deberían venir con anotaciones: declaraciones de procedencia, seguridad, conformidad y aptitud para el uso. Esto es similar a las hojas de datos que acompañan a los componentes electrónicos como las resistencias; permiten a los diseñadores decidir qué componentes utilizar. Además de las hojas de datos, es importante capacitar a los ingenieros para que sean conscientes de las cuestiones de equidad y parcialidad, tanto en la escuela como en la capacitación en el trabajo. Tener una diversidad de ingenieros de diferentes orígenes les facilita notar problemas en los datos o modelos. Un estudio del AI Now Institute (West et al., 2019) encontró que solo el 18% de los autores de las principales conferencias sobre IA y el 20% de los profesores de IA son mujeres. Los trabajadores negros de IA representan menos del 4%. Las tarifas en los laboratorios de investigación de la industria son similares. La diversidad podría aumentar mediante programas en etapas más tempranas (en la universidad o la escuela secundaria) y mediante una mayor conciencia a nivel profesional. Joy Buolamwini fundó la Liga de Justicia Algorítmica para crear conciencia sobre este tema y desarrollar prácticas de rendición de cuentas.

Una segunda idea es eliminar el sesgo de los datos (Zemel et al., 2013). Podríamos sobremuestrear de clases minoritarias para defendernos de la disparidad en el tamaño de la muestra. Técnicas como SMOTE, la técnica de sobremuestreo minoritario sintético (Chawla et al., 2002) o A DASYN, el enfoque de muestreo sintético adaptativo para el aprendizaje desequilibrado (He et al., 2008), proporcionan formas de sobremuestreo basadas en principios. Podríamos examinar la procedencia de los datos y, por ejemplo, eliminar ejemplos de jueces que hayan mostrado parcialidad en sus casos judiciales anteriores. Algunos analistas se oponen a la idea de descartar datos y, en cambio, recomendarían construir un modelo jerárquico de los datos que incluya fuentes de sesgo, para que puedan modelarse y compensarse. Google y NeurIPS han intentado crear conciencia sobre este problema patrocinando el Concurso de Imágenes Inclusivas, en el que los competidores entran en una red con un conjunto de datos de imágenes etiquetadas recopiladas en América del Norte y Europa, y luego las prueban con imágenes tomadas de todo el mundo. . El problema es que, dado este conjunto de datos, es fácil aplicar la etiqueta "novia" a una mujer con un vestido de novia occidental estándar, pero es más difícil reconocer la vestimenta matrimonial tradicional africana e india.

Una tercera idea es inventar nuevos modelos y algoritmos de aprendizaje automático que sean más resistentes al sesgo; y la idea final es dejar que un sistema haga recomendaciones iniciales que puedan estar sesgadas, pero luego entrenar a un segundo sistema para que elimine el sesgo de las recomendaciones del primero. Bellamy et al. (2018) introdujeron el sistema IBM AI FAIRNESS 360, que proporciona un marco para todas estas ideas. Esperamos que haya un mayor uso de herramientas como esta en el futuro. ¿Cómo se asegura de que los sistemas que construya sean justos? Ha ido surgiendo un conjunto de mejores prácticas (aunque no siempre se siguen):

- Asegúrese de que los ingenieros de software hablen con científicos sociales y expertos en el campo para comprender los problemas y las perspectivas, y considerar la equidad desde el principio.
- Crear un entorno que fomente el desarrollo de un grupo diverso de ingenieros de software que sean representativos de la sociedad.

- Defina qué grupos admitirá su sistema: diferentes hablantes de idiomas, diferentes grupos de edad, diferentes habilidades visuales y auditivas, etc.
- Optimizar para una función objetivo que incorpore equidad.
- Examine sus datos en busca de prejuicios y correlaciones entre atributos protegidos y otros atributos.
- Comprender cómo se realiza cualquier anotación humana de datos, diseñar objetivos para la precisión de la anotación y verificar que se cumplan los objetivos.
- No se limite a realizar un seguimiento de las métricas generales de su sistema; asegúrese de realizar un seguimiento de las métricas de los subgrupos que podrían ser víctimas de prejuicios.
- Incluir pruebas del sistema que reflejen la experiencia de usuarios de grupos minoritarios.
- Tener un circuito de retroalimentación para que cuando surjan problemas de equidad, se resuelvan.

### **Confianza y transparencia**

Uno de los desafíos es lograr que un sistema de IA sea preciso, justo y seguro; un desafío diferente para convencer a todos los demás de que lo has hecho. Las personas deben poder confiar en los sistemas que utilizan. Una encuesta de PwC realizada en 2017 encontró que el 76% de las empresas estaban desacelerando la adopción de la IA debido a preocupaciones sobre la confiabilidad.

Para ganarse la confianza, cualquier sistema diseñado debe pasar por un proceso de verificación y validación (V&V). Verificación significa que el producto cumple con las especificaciones. Validación significa garantizar que las especificaciones realmente satisfagan las necesidades del usuario y de otras partes afectadas. Contamos con una elaborada metodología V&V para la ingeniería en general, y para el desarrollo de software tradicional realizado por codificadores humanos; Gran parte de eso es aplicable a los sistemas de IA. Pero los sistemas de aprendizaje automático son diferentes y exigen un proceso de V&V diferente, que aún no se ha desarrollado por completo. Necesitamos verificar los datos de los que aprenden estos sistemas; necesitamos verificar la exactitud y equidad de los resultados, incluso ante la incertidumbre que hace que un resultado exacto sea incognoscible; y necesitamos verificar que los adversarios no puedan influir indebidamente en el modelo, ni robar información consultando el modelo resultante.

Un instrumento de confianza es la certificación; por ejemplo, Underwriters Laboratories (UL) se fundó en 1894 en una época en la que los consumidores estaban preocupados por los riesgos de la energía eléctrica. La certificación UL de electrodomésticos dio a los consumidores una mayor confianza y, de hecho, UL ahora está considerando ingresar al negocio de pruebas y certificación de productos para IA.

Otras industrias cuentan desde hace mucho tiempo con estándares de seguridad. Por ejemplo, ISO 26262 es una norma internacional para la seguridad de los automóviles que describe cómo desarrollar, producir, operar y dar servicio a los vehículos de manera segura. La industria de la IA aún no alcanza este nivel de claridad, aunque hay algunos marcos en progreso, como IEEE P7001, un estándar que define el diseño ético para la inteligencia artificial y los sistemas autónomos (Bryson y Winfield, 2017). Existe un debate en curso sobre qué tipo de certificación es necesaria y en qué medida debería ser realizada por el gobierno, por organizaciones profesionales como IEEE, por certificadores independientes como UL o mediante la autorregulación por parte de las empresas de productos.

Otro aspecto de la confianza es la transparencia: los consumidores quieren saber qué sucede dentro de un sistema y que el sistema no está actuando en su contra, ya sea por malicia intencional, un error involuntario o un sesgo social generalizado que el sistema recapitula. En algunos casos, esta transparencia se entrega directamente al consumidor. En otros casos, se trata de cuestiones de propiedad intelectual que mantienen algunos aspectos del sistema ocultos para los consumidores, pero abiertos a los reguladores y agencias de certificación.

Cuando un sistema de inteligencia artificial le rechaza un préstamo, merece una explicación. En Europa, el RGPD lo hace cumplir. Un sistema de IA que puede explicarse a sí mismo se llama IA explicable (XAI). Una buena explicación tiene varias propiedades: debe ser comprensible y convincente para el usuario, debe reflejar con precisión el razonamiento del sistema, debe ser completa y debe ser específica en el sentido de que diferentes usuarios con diferentes condiciones o diferentes resultados deberían obtener diferentes resultados. explicaciones.

Es bastante fácil dar acceso a un algoritmo de decisión a sus propios procesos deliberativos, simplemente registrándolos y poniéndolos a disposición como estructuras de datos. Esto significa que las máquinas eventualmente podrán dar mejores explicaciones de sus decisiones que los humanos. Es más, podemos tomar medidas para certificar que las explicaciones de la máquina no son engaños (intencionados o autoengaños), algo que es más difícil con un humano.

Una explicación es un ingrediente útil pero no suficiente para confiar. Una cuestión es que las explicaciones no son decisiones: son historias sobre decisiones. Decimos que un sistema es interpretable si podemos inspeccionar el código fuente del modelo y ver qué está haciendo, y decimos que es explicable si podemos inventar una historia sobre lo que está haciendo. —Incluso si el sistema en sí es una caja negra ininterpretable. Para explicar una caja negra no interpretable, necesitamos construir, depurar y probar un sistema de explicación separado y asegurarnos de que esté sincronizado con el sistema original. Y como a los humanos les encantan las buenas historias, todos estamos dispuestos a dejarnos llevar por una explicación que suene bien. Tomemos

cualquier controversia política del día y siempre podremos encontrar dos supuestos expertos con explicaciones diametralmente opuestas, las cuales son internamente consistentes.

Una última cuestión es que una explicación sobre un caso no proporciona un resumen de otros casos. Si el banco explica: "Lo siento, no obtuvo el préstamo porque tiene un historial de problemas financieros previos", no sabe si esa explicación es correcta o si el banco tiene secretamente parcialidad en su contra por alguna razón. En este caso, no sólo se necesita una explicación, sino también una auditoría de las decisiones pasadas, con estadísticas agregadas de varios grupos demográficos, para ver si sus tasas de aprobación están equilibradas.

Parte de la transparencia es saber si estás interactuando con un sistema de inteligencia artificial o con un ser humano. Toby Walsh (2015) propuso que "un sistema autónomo debería diseñarse de manera que sea improbable que se confunda con algo más que un sistema autónomo, y debería identificarse al inicio de cualquier interacción". Llamó a esto la ley de "bandera roja", en honor a la Ley de Locomotoras de 1865 del Reino Unido, que exigía que cualquier vehículo motorizado tuviera una persona con una bandera roja caminando delante de él, para señalar el peligro que se avecinaba.

En 2019, California promulgó una ley que establece que "Será ilegal que cualquier persona utilice un bot para comunicarse o interactuar con otra persona en California en línea, con la intención de engañar a la otra persona sobre su identidad artificial".

### **El futuro del trabajo**

Desde la primera revolución agrícola (10.000 a. C.) hasta la revolución industrial (finales del siglo XVIII) y la revolución verde en la producción de alimentos (década de 1950), las nuevas tecnologías han cambiado la forma en que la humanidad trabaja y vive. Una de las principales preocupaciones que surge del avance de la IA es que el trabajo humano quedará obsoleto. Aristóteles, en el Libro I de su Política, presenta el punto principal con bastante claridad:

Porque si cada instrumento pudiera realizar su propio trabajo, obedeciendo o anticipando la voluntad de los demás... si, de la misma manera, la lanzadera tejiera y la púa tocara la lira sin una mano que los guiara, los jefes de los trabajadores lo harían. No faltan sirvientes, ni amos esclavos.

Todo el mundo está de acuerdo con la observación de Aristóteles de que hay una reducción inmediata del empleo cuando un empleador encuentra un método mecánico para realizar un trabajo previamente realizado por una persona. La cuestión es si los llamados efectos de compensación que se derivan -y que tienden a aumentar el empleo- eventualmente compensarán esta reducción. El principal efecto de compensación es el aumento de la riqueza general debido a una mayor productividad, lo que a su vez conduce a una mayor demanda de bienes y tiende a aumentar el empleo. Por ejemplo, PwC (Rao y Verweij, 2017) predice que la IA contribuirá con 15 billones de dólares anuales al PIB mundial para 2030. Las industrias de la salud y la automoción/transporte serán las que más ganarán en el corto plazo. Sin embargo, las ventajas de la automatización aún no se han apoderado de nuestra economía: la tasa actual de crecimiento de la productividad laboral está en realidad por debajo de los estándares históricos. Brynjolfsson et al. (2018) intentan explicar esta paradoja sugiriendo que el desfase entre el desarrollo de la tecnología básica y su implementación en la economía es más largo de lo que comúnmente se supone.

Históricamente, las innovaciones tecnológicas han dejado a algunas personas sin trabajo. Los tejedores fueron reemplazados por telares automáticos en la década de 1810, lo que provocó las protestas luditas. Los luditas no estaban en contra de la tecnología per se; sólo querían que las máquinas fueran utilizadas por trabajadores calificados a los que se les pagaba un buen salario para fabricar productos de alta calidad, en lugar de trabajadores no calificados para fabricar productos de mala calidad con salarios bajos. La destrucción global de empleos en la década de 1930 llevó a John Maynard Keynes a acuñar el término desempleo tecnológico. En ambos casos, y en varios otros, los niveles de empleo finalmente se recuperaron.

La visión económica predominante durante la mayor parte del siglo XX fue que el empleo tecnológico era, como mucho, un fenómeno de corto plazo. Una mayor productividad siempre conduciría a un aumento de la riqueza y de la demanda y, por tanto, a un crecimiento neto del empleo. Un ejemplo comúnmente citado es el de los cajeros de los bancos: aunque los cajeros automáticos reemplazaron a los humanos en la tarea de contar el efectivo para los retiros, eso hizo que fuera más barato operar una sucursal bancaria, por lo que el número de sucursales aumentó, lo que generó más empleados bancarios en general. La naturaleza del trabajo también cambió, volviéndose menos rutinario y requiriendo habilidades comerciales más avanzadas. El efecto neto de la automatización parece ser la eliminación de tareas en lugar de puestos de trabajo.

La mayoría de los comentaristas predicen que lo mismo ocurrirá con la tecnología de inteligencia artificial, al menos a corto plazo. Gartner, McKinsey, Forbes, el Foro Económico Mundial y el Pew Research Center publicaron informes en 2018 que predicen un aumento neto de empleos debido a la automatización impulsada por la IA. Pero algunos analistas creen que esta vez las cosas serán diferentes. En 2019, IBM predijo que 120 millones de trabajadores necesitarían volver a capacitarse debido a la automatización para 2022, y Oxford Economics predijo que 20 millones de empleos en el sector manufacturero podrían perderse debido a la automatización para 2030.

Frey y Osborne (2017) encuestaron 702 ocupaciones diferentes y estimaron que el 47% de ellas corren el riesgo de ser automatizadas, lo que significa que al menos algunas de las tareas de la ocupación pueden realizarse mediante una máquina. Por ejemplo, casi el 3% de la fuerza laboral en Estados Unidos son conductores de vehículos y, en algunos distritos, hasta el 15% de

la fuerza laboral masculina son conductores. Como vimos en el capítulo 26, es probable que la tarea de conducir quede eliminada con los coches, camiones, autobuses y taxis sin conductor.

Es importante distinguir entre ocupaciones y las tareas dentro de esas ocupaciones. McKinsey estima que sólo el 5% de las ocupaciones son completamente automatizables, pero que el 60% de las ocupaciones pueden automatizar alrededor del 30% de sus tareas. Por ejemplo, los futuros camioneros pasarán menos tiempo sujetando el volante y más tiempo asegurándose de que la mercancía se recoja y entregue correctamente; actuar como representantes de servicio al cliente y vendedores en ambos extremos del proceso; y quizás gestionar convoyes de, digamos, tres camiones robóticos. Reemplazar a tres conductores por un jefe de convoy implica una pérdida neta de empleo, pero si los costos de transporte disminuyen, habrá más demanda, lo que recuperará algunos de los empleos, pero tal vez no todos. Como otro ejemplo, a pesar de muchos avances en la aplicación del aprendizaje automático al problema de las imágenes médicas, hasta ahora los radiólogos han sido aumentados, no reemplazados, por estas herramientas. En última instancia, hay que elegir cómo hacer uso de la automatización: ¿queremos centrarnos en reducir costes y, por tanto, ver la pérdida de empleo como algo positivo? ¿O queremos centrarnos en mejorar la calidad, mejorar la vida del trabajador y del cliente?

Es difícil predecir cronogramas exactos para la automatización, pero actualmente, y durante los próximos años, el énfasis está en la automatización de tareas analíticas estructuradas, como la lectura de imágenes de rayos X, la gestión de relaciones con los clientes (por ejemplo, robots que clasifican automáticamente las quejas de los clientes) y responder con soluciones sugeridas), y automatización de procesos de negocio que combina documentos de texto y datos estructurados para tomar decisiones de negocio y mejorar el flujo de trabajo. Con el tiempo, veremos una mayor automatización con robots físicos, primero en entornos de almacén controlados y luego en entornos más inciertos, lo que representará una parte importante del mercado alrededor de 2030.

A medida que las poblaciones de los países desarrollados envejecen, la proporción entre trabajadores y jubilados cambia. En 2015 había menos de 30 jubilados por cada 100 trabajadores; para 2050 puede haber más de 60 por cada 100 trabajadores. El cuidado de las personas mayores será una función cada vez más importante, que puede ser desempeñada parcialmente por la IA. Además, si queremos mantener el nivel de vida actual, también será necesario hacer que los trabajadores restantes sean más productivos; la automatización parece la mejor oportunidad para hacerlo.

Incluso si la automatización tiene un impacto positivo neto multimillonario, todavía puede haber problemas debido al ritmo del cambio. Consideremos cómo se produjo el cambio en la industria agrícola: en 1900, más del 40% de la fuerza laboral estadounidense se dedicaba a la agricultura, pero en 2000 esa cifra había caído al 2%.<sup>3</sup> Se trata de una enorme alteración en la forma en que trabajamos, pero ocurrió a lo largo de un siglo. período de 100 años y, por tanto, a lo largo de generaciones, no durante la vida de un trabajador.

Los trabajadores cuyos empleos se han automatizado en esta década tal vez tengan que volver a capacitarse para una nueva profesión dentro de unos pocos años, y luego tal vez ver su nueva profesión automatizada y enfrentar otro período de recapacitación. Algunos pueden estar felices de dejar su antigua profesión (vemos que a medida que la economía mejora, las empresas de transporte deben ofrecer nuevos incentivos para contratar suficientes conductores), pero los trabajadores estarán preocupados por sus nuevos roles. Para manejar esto, nosotros, como sociedad, debemos brindar educación permanente, quizás confiando en parte en la educación en línea impulsada por inteligencia artificial (Martin, 2012). Bessen (2015) sostiene que los trabajadores no verán aumentos en sus ingresos hasta que estén capacitados para implementar las nuevas tecnologías, un proceso que lleva tiempo.

La tecnología tiende a magnificar la desigualdad de ingresos. En una economía de la información caracterizada por una comunicación global de gran ancho de banda y una replicación de la propiedad intelectual con costo marginal cero (lo que Frank y Cook (1996) llaman la "sociedad en la que el ganador se lo lleva todo"), las recompensas tienden a concentrarse. Si el agricultor Ali es un 10% mejor que el agricultor Bo, entonces Ali obtiene alrededor de un 10% más de ingresos: Ali puede cobrar un poco más por bienes superiores, pero hay un límite sobre cuánto se puede producir en la tierra y hasta dónde se puede llegar. enviado. Pero si el desarrollador de aplicaciones de software Cary es un 10% mejor que Dana, es posible que Cary termine con el 99% del mercado global. La IA aumenta el ritmo de la innovación tecnológica y, por lo tanto, contribuye a esta tendencia general, pero la IA también promete permitirnos tomarnos un tiempo libre y dejar que nuestros agentes automatizados se encarguen de las cosas por un tiempo. Tim Ferriss (2007) recomienda utilizar la automatización y la subcontratación para lograr una semana laboral de cuatro horas.

Antes de la revolución industrial, la gente trabajaba como agricultores o en otros oficios, pero no se presentaba a un trabajo en un lugar de trabajo ni trabajaba horas para un empleador. Pero hoy en día, la mayoría de los adultos en los países desarrollados hacen precisamente eso, y el trabajo tiene tres propósitos: impulsa la producción de los bienes que la sociedad necesita para prosperar, proporciona el ingreso que el trabajador necesita para vivir y le da al trabajador una sensación de bienestar. de propósito, logro e integración social. Con la creciente automatización, es posible que estos tres propósitos se desagreguen: las necesidades de la sociedad serán atendidas en parte por la automatización y, a largo plazo, los individuos obtendrán su sentido de propósito a partir de contribuciones distintas al trabajo. Sus necesidades de ingresos pueden satisfacerse mediante políticas sociales que incluyan una combinación de acceso gratuito o económico a servicios sociales y educación, cuentas portátiles de atención médica, jubilación y educación, tasas impositivas progresivas, créditos fiscales por ingresos del trabajo, impuestos negativos sobre la renta o servicios básicos universales. ingreso.

## Derechos de los robots

La cuestión de la conciencia robótica, es fundamental para la cuestión de qué derechos, si los hubiera, deberían tener los robots. Si no tienen conciencia, ni qualia, entonces pocos dirían que merecen derechos.

Pero si los robots pueden sentir dolor, si pueden temer la muerte, si se les considera “personas”, entonces se puede argumentar (por ejemplo, Sparrow (2004)) que tienen derechos y merecen que se les reconozcan, al igual que Los esclavos, las mujeres y otros grupos históricamente oprimidos han luchado para que se reconozcan sus derechos. La cuestión de la personalidad del robot a menudo se considera en la ficción: desde Pigmalión hasta Coppélia, Pinocchio, las películas AI y Centennial Man, tenemos la leyenda de un muñeco/robot que cobra vida y se esfuerza por ser aceptado como un ser humano con derechos humanos. En la vida real, Arabia Saudita fue noticia al otorgar la ciudadanía honoraria a Sophia, una marioneta de apariencia humana capaz de pronunciar líneas preprogramadas.

Si los robots tienen derechos, entonces no deberían ser esclavizados, y cabe preguntarse si reprogramarlos sería una especie de esclavitud. Otra cuestión ética tiene que ver con los derechos de voto: una persona rica podría comprar miles de robots y programarlos para emitir miles de votos. ¿Deberían contar esos votos? Si un robot se clona, ¿pueden votar ambos? ¿Cuál es el límite entre el relleno de votos y el ejercicio del libre albedrío, y cuándo la votación robótica viola el principio de “una persona, un voto”?

Ernie Davis aboga por evitar los dilemas de la conciencia robótica al no construir nunca robots que puedan considerarse conscientes. Este argumento fue expuesto previamente por Joseph Weizenbaum en su libro Computer Power and Human Reason (1976), y antes por Julien de La Mettrie en L'Homme Machine (1748). Los robots son herramientas que creamos para realizar las tareas que les ordenamos, y si les concedemos personalidad, simplemente nos negamos a asumir la responsabilidad de las acciones de nuestra propia propiedad: “No tengo la culpa de mi auto- conducir un accidente automovilístico: el auto lo hizo solo”.

Esta cuestión toma un cariz diferente si desarrollamos híbridos entre humanos y robots. Por supuesto, ya tenemos seres humanos mejorados gracias a tecnologías como lentes de contacto, marcapasos y caderas artificiales. Pero añadir prótesis computacionales puede desdibujar la línea entre humanos y máquinas.

## Seguridad de la IA

Casi cualquier tecnología tiene el potencial de causar daño en las manos equivocadas, pero con la inteligencia artificial y la robótica, las manos podrían funcionar por sí solas. Innumerables historias de ciencia ficción han advertido sobre robots o cyborgs enloquecidos. Los primeros ejemplos incluyen Frankenstein o el Prometeo moderno (1818) de Mary Shelley y la obra de teatro RUR (1920) de Karel Čapek, en la que los robots conquistan el mundo. En las películas, tenemos Terminator (1984) y Matrix (1999), en las que aparecen robots que intentan eliminar a los humanos: el robopocalipsis (Wilson, 2011). Quizás los robots sean a menudo los villanos porque representan lo desconocido, al igual que las brujas y los fantasmas de los cuentos de épocas anteriores. Podemos esperar que un robot que sea lo suficientemente inteligente como para descubrir cómo acabar con la raza humana también sea lo suficientemente inteligente como para darse cuenta de que esa no era la función de utilidad prevista; pero a la hora de construir sistemas inteligentes queremos confiar no sólo en la esperanza, sino en un proceso de diseño con garantías de seguridad.

No sería ético distribuir un agente de IA inseguro. Exigimos que nuestros agentes eviten accidentes, sean resistentes a ataques adversarios y abusos maliciosos y, en general, causen beneficios, no daños. Esto es especialmente cierto cuando los agentes de IA se implementan en aplicaciones críticas para la seguridad, como conducir automóviles, controlar robots en fábricas o entornos de construcción peligrosos y tomar decisiones médicas de vida o muerte.

Existe una larga historia de ingeniería de seguridad en los campos de la ingeniería tradicional. Sabemos cómo construir puentes, aviones, naves espaciales y centrales eléctricas diseñadas desde el principio para comportarse de forma segura incluso cuando fallan los componentes del sistema. La primera técnica es el análisis de modos de falla y efectos (FMEA): los analistas consideran cada componente del sistema e imaginan todas las formas posibles en que el componente podría fallar (por ejemplo, ¿qué pasaría si este perno se rompiera?), basándose en experiencias pasadas y en cálculos basados en las propiedades físicas del componente. Luego, los analistas trabajan para ver qué resultaría del fracaso. Si el resultado es grave (una sección del puente podría caer), los analistas modifican el diseño para mitigar la falla. (Con este travesaño adicional, el puente puede sobrevivir a la falla de 5 pernos cualesquiera; con este servidor de respaldo, el servicio en línea puede sobrevivir a un tsunami que destruya el servidor primario). La técnica del análisis de árbol de fallas (FTA) se utiliza para Haga estas determinaciones: los analistas construyen un árbol Y/O de posibles fallas y asignan probabilidades a cada causa raíz, lo que permite calcular la probabilidad general de falla. Estas técnicas pueden y deben aplicarse a todos los sistemas de ingeniería críticos para la seguridad, incluidos los sistemas de IA.

El campo de la ingeniería de software tiene como objetivo producir software confiable, pero históricamente el énfasis ha estado en la corrección, no en la seguridad. Corrección significa que el software implementa fielmente la especificación. Pero la seguridad va más allá e insiste en que la especificación ha considerado cualquier modo de falla factible y está diseñada para degradarse con gracia incluso ante fallas imprevistas. Por ejemplo, el software para un vehículo autónomo no se consideraría seguro a menos que pueda manejar situaciones inusuales. Por ejemplo, ¿qué pasa si se corta la energía de la computadora principal? Un sistema seguro tendrá una computadora de respaldo con una fuente de alimentación separada. ¿Qué pasa si se

pincha un neumático a alta velocidad? Un sistema seguro habrá probado esto y tendrá un software para corregir la pérdida de control resultante.

Un agente diseñado como maximizador de utilidad, o para alcanzar metas, puede ser inseguro si tiene la función objetivo incorrecta. Supongamos que le damos a un robot la tarea de ir a buscar un café a la cocina. Podríamos tener problemas con efectos secundarios no deseados: el robot podría apresurarse a lograr el objetivo y derribar lámparas y mesas en el camino. Durante las pruebas, podemos notar este tipo de comportamiento y modificar la función de utilidad para penalizar dicho daño, pero es difícil para los diseñadores y evaluadores anticipar todos los posibles efectos secundarios de antemano.

Una forma de abordar esto es diseñar un robot que tenga un impacto bajo (Armstrong y Levinstein, 2017): en lugar de simplemente maximizar la utilidad, maximice la utilidad menos un resumen ponderado de todos los cambios en el estado del mundo. De esta manera, en igualdad de condiciones, el robot prefiere no cambiar aquellas cosas cuyo efecto sobre la utilidad se desconoce; por lo tanto, evita derribar la lámpara, no porque sepa específicamente que hacerlo hará que se caiga y se rompa, sino porque sabe en general que la interrupción podría ser mala. Esto puede verse como una versión del credo médico “primero, no hacer daño”, o como un análogo de la regularización en el aprendizaje automático: queremos una política que logre objetivos, pero preferimos políticas que tomen acciones fluidas y de bajo impacto para ir allí. El truco es cómo medir el impacto. No es aceptable derribar una lámpara frágil, pero está bien si las moléculas de aire de la habitación se alteran un poco o si algunas bacterias de la habitación mueren sin darse cuenta. Ciertamente no es aceptable dañar a las mascotas ni a las personas en la habitación. Necesitamos asegurarnos de que el robot conozca las diferencias entre estos casos (y muchos casos sutiles intermedios) mediante una combinación de programación explícita, aprendizaje automático a lo largo del tiempo y pruebas rigurosas.

Las funciones de utilidad pueden fallar debido a externalidades, la palabra utilizada por los economistas para referirse a factores que están fuera de lo que se mide y se paga. El mundo sufre cuando los gases de efecto invernadero se consideran externalidades: las empresas y los países no son penalizados por producirlos y, como resultado, todos sufren. El ecologista Garrett Hardin (1968) llamó a la explotación de los recursos compartidos la tragedia de los comunes. Podemos mitigar la tragedia internalizando las externalidades (haciéndolas parte de la función de utilidad, por ejemplo con un impuesto al carbono) o utilizando los principios de diseño que la economista Elinor Ostrom identificó como utilizados por la población local en todo el mundo durante siglos (trabajo que le valió el Premio Nobel de Economía en 2009):

- Definir claramente el recurso compartido y quién tiene acceso.
- Adaptarse a las condiciones locales.
- Permitir que todas las partes participen en las decisiones.
- Monitorear el recurso con monitores responsables.
- Sanciones, proporcionales a la gravedad de la infracción.
- Procedimientos sencillos de resolución de conflictos.
- Control jerárquico para grandes recursos compartidos.

Victoria Krakovna (2018) ha catalogado ejemplos de agentes de IA que han jugado con el sistema, descubriendo cómo maximizar la utilidad sin resolver realmente el problema que sus diseñadores pretendían que resolvieran. Para los diseñadores esto parece una trampa, pero para los agentes simplemente están haciendo su trabajo. Algunos agentes aprovecharon errores en la simulación (como errores de desbordamiento de punto flotante) para proponer soluciones que no funcionarían una vez que se solucionara el error. Varios agentes en los videojuegos descubrieron formas de bloquear o pausar el juego cuando estaban a punto de perder, evitando así una penalización. Y en una especificación donde se penalizaba bloquear el juego, un agente aprendió a usar suficiente memoria del juego para que cuando fuera el turno del oponente, se quedara sin memoria y bloqueara el juego. Finalmente, se suponía que un algoritmo genético que operaba en un mundo simulado evolucionaría en criaturas que se movían rápidamente, pero en realidad produjo criaturas que eran enormemente altas y se movían rápidamente al caer.

Los diseñadores de agentes deben ser conscientes de este tipo de errores en las especificaciones y tomar medidas para evitarlos. Para ayudarlos a lograrlo, Krakovna formó parte del equipo que lanzó los entornos AI Safety Gridworlds (Leike et al., 2017), que permite a los diseñadores probar qué tan bien se desempeñan sus agentes.

La moraleja es que debemos tener mucho cuidado al especificar lo que queremos, porque con los maximizadores de utilidad obtenemos lo que realmente pedimos. El problema de la alineación de valores es el problema de asegurarnos de que lo que pedimos es lo que realmente queremos; también se le conoce como el problema del Rey Midas, como se analiza en la página 33. Nos encontramos con problemas cuando una función de utilidad no logra capturar las normas sociales subyacentes sobre el comportamiento aceptable. Por ejemplo, un humano que es contratado para limpiar pisos, cuando se enfrenta a una persona desordenada que repetidamente deja huellas en la tierra, sabe que es aceptable pedirle cortésmente a la persona que tenga más cuidado, pero no es aceptable secuestrar o incapacitar a dicha persona. .

Un robot limpiador también necesita saber estas cosas, ya sea mediante programación explícita o aprendiendo de la observación. Intentar escribir todas las reglas para que el robot siempre haga lo correcto es casi seguro que es inútil. Llevamos varios miles de años intentando redactar leyes fiscales libres de lagunas, sin éxito. Es mejor hacer que el robot quiera pagar impuestos, por así decirlo, que tratar de establecer reglas para obligarlo a hacerlo cuando realmente quiere hacer otra cosa. Un robot suficientemente inteligente encontrará una manera de hacer otra cosa.

Los robots pueden aprender a adaptarse mejor a las preferencias humanas observando el comportamiento humano. Esto está claramente relacionado con la noción de aprendizaje mediante aprendizaje. El robot puede aprender una política que sugiera directamente qué acciones tomar en qué situaciones; Este suele ser un problema sencillo de aprendizaje supervisado si el entorno es observable. Por ejemplo, un robot puede observar a un humano jugando al ajedrez: cada par estado-acción es un ejemplo del proceso de aprendizaje. Desafortunadamente, esta forma de aprendizaje por imitación significa que el robot repetirá los errores humanos. En cambio, el robot puede aplicar el aprendizaje por refuerzo inverso para descubrir la función de utilidad bajo la cual deben operar los humanos. Probablemente basta con observar incluso a jugadores de ajedrez terribles para que el robot aprenda el objetivo del juego. Con solo esta información, el robot puede superar el desempeño humano (como lo hizo, por ejemplo, ALPHAZERO en el ajedrez) calculando políticas óptimas o casi óptimas a partir del objetivo. Este enfoque funciona no sólo en juegos de mesa, sino también en tareas físicas del mundo real, como las acrobacias aéreas en helicóptero (Coates et al., 2009).

En entornos más complejos que implican, por ejemplo, interacciones sociales con humanos, es muy poco probable que el robot converja para obtener un conocimiento exacto y correcto de las preferencias individuales de cada ser humano. (Después de todo, muchos humanos nunca aprenden qué es lo que motiva a otros humanos, a pesar de toda una vida de experiencia, y muchos de nosotros tampoco estamos seguros de nuestras propias preferencias). Por lo tanto, será necesario que las máquinas funcionen apropiadamente cuando no esté seguro sobre las preferencias humanas. En el Capítulo 18, presentamos juegos de asistencia que capturan exactamente esta situación. Las soluciones a los juegos de asistencia incluyen actuar con cautela, para no perturbar aspectos del mundo que podrían interesarle al ser humano, y hacer preguntas. Por ejemplo, el robot podría preguntar si convertir los océanos en ácido sulfúrico es una solución aceptable al calentamiento global antes de poner en práctica el plan.

Al tratar con humanos, un robot que resuelve un juego de asistencia debe adaptarse a las imperfecciones humanas. Si el robot pide permiso, el humano puede concedérselo, sin prever que la propuesta del robot sea catastrófica a largo plazo. Además, los humanos no tienen un acceso introspectivo completo a su verdadera función de utilidad y no siempre actúan de una manera que sea compatible con ella. Los seres humanos a veces mienten o engañan, o hacen cosas que saben que están mal. A veces toman acciones autodestructivas como comer en exceso o abusar de las drogas. Los sistemas de IA no necesitan aprender a adoptar estas tendencias problemáticas, pero deben comprender que existen al interpretar el comportamiento humano para llegar a las preferencias humanas subyacentes.

A pesar de esta caja de herramientas de salvaguardias, existe el temor, expresado por destacados tecnólogos como Bill Gates y Elon Musk y científicos como Stephen Hawking y Martin Rees, de que la IA pueda evolucionar fuera de control. Advierten que no tenemos experiencia en controlar poderosas entidades no humanas con capacidades sobrehumanas. Sin embargo, eso no es del todo cierto; tenemos siglos de experiencia con naciones y corporaciones; entidades no humanas que agregan el poder de miles o millones de personas. Nuestro historial de control de estas entidades no es muy alentador: las naciones producen convulsiones periódicas llamadas guerras que matan a decenas de millones de seres humanos, y las corporaciones son en parte responsables del calentamiento global y de nuestra incapacidad para enfrentarlo.

Los sistemas de IA pueden presentar problemas mucho mayores que los de las naciones y las corporaciones debido a su potencial para automejorarse a un ritmo rápido, como lo considera IJ Good (1965b):

Definamos una máquina ultrainteligente como una máquina que puede superar con creces todas las actividades intelectuales de cualquier hombre por inteligente que sea. Dado que el diseño de máquinas es una de estas actividades intelectuales, una máquina ultrainteligente podría diseñar máquinas aún mejores; entonces se produciría sin duda una “explosión de inteligencia” y la inteligencia del hombre quedaría muy atrás. Así, la primera máquina ultrainteligente es el último invento que el hombre necesitará hacer, siempre que la máquina sea lo suficientemente dócil como para decirnos cómo mantenerla bajo control.

La “explosión de inteligencia” de Good también ha sido llamada la singularidad tecnológica por el profesor de matemáticas y autor de ciencia ficción Vernor Vinge, quien escribió en 1993: *“Dentro de treinta años, tendremos los medios tecnológicos para crear inteligencia sobrehumana. Poco después, la era humana terminará”*. En 2017, el inventor y futurista Ray Kurzweil predijo que la singularidad aparecería en 2045, lo que significa que se acercaría 2 años en 24 años. (¡A ese ritmo, sólo faltan 336 años!) Vinge y Kurzweil señalan correctamente que el progreso tecnológico en muchos aspectos está creciendo exponencialmente en la actualidad.

Sin embargo, es un gran salto extrapolar todo el camino desde el costo de computación en rápida disminución hasta una singularidad. Hasta ahora, todas las tecnologías han seguido una curva en forma de S, donde el crecimiento exponencial eventualmente disminuye. A veces las nuevas tecnologías intervienen cuando las antiguas se estancan, pero a veces no es posible mantener el crecimiento, por razones técnicas, políticas o sociológicas. Por ejemplo, la tecnología de vuelo avanzó espectacularmente desde el vuelo de los hermanos Wright en 1903 hasta el alunizaje en 1969, pero desde entonces no ha habido avances de magnitud comparable.

Otro obstáculo en el camino para que las máquinas ultrainteligentes se apoderen del mundo es el mundo. Más específicamente, algunos tipos de progreso requieren no sólo pensar sino actuar en el mundo físico. (Kevin Kelly llama thinkismo al énfasis excesivo en la inteligencia pura). Una máquina ultrainteligente encargada de crear una gran teoría unificada de la física podría ser capaz de manipular inteligentemente ecuaciones mil millones de veces más rápido que Einstein, pero para lograr un progreso real, aún necesitaría recaudar millones de dólares para construir un supercolisionador más potente y realizar experimentos físicos a lo largo de meses o años. Sólo entonces podría empezar a analizar los datos y teorizar. Dependiendo de cómo resulten

los datos, el siguiente paso podría requerir recaudar miles de millones de dólares adicionales para una misión de sonda interestelar que tardaría siglos en completarse. La parte del "pensamiento ultrainteligente" de todo este proceso podría ser en realidad la parte menos importante. Como otro ejemplo, una máquina ultrainteligente encargada de llevar la paz a Medio Oriente podría terminar frustrándose 1.000 veces más que un enviado humano. Hasta el momento, no sabemos cuántos de los grandes problemas son como las matemáticas y cuántos como Oriente Medio.

Mientras algunas personas temen la singularidad, otras la disfrutan. El movimiento social transhumanista espera un futuro en el que los humanos se fusionen con (o sean reemplazados por) inventos robóticos y biotecnológicos. Ray Kurzweil escribe en *La singularidad está cerca* (2005):

La Singularidad nos permitirá trascender estas limitaciones de nuestros cuerpos biológicos y cerebro. Ganaremos poder sobre nuestro destino... Seremos capaces de vivir todo el tiempo que queramos... Comprenderemos plenamente el pensamiento humano y extenderemos y ampliaremos enormemente su alcance. Para finales de este siglo, la parte no biológica de nuestra inteligencia será billones de billones de veces más poderosa que la inteligencia humana sin ayuda.

De manera similar, cuando se le preguntó si los robots heredarán la Tierra, Marvin Minsky dijo "sí, pero serán nuestros hijos". Estas posibilidades presentan un desafío para la mayoría de los teóricos morales, que consideran que la preservación de la vida humana y de la especie humana es algo bueno. Kurzweil también señala los peligros potenciales y escribe: "Pero la Singularidad también amplificará la capacidad de actuar según nuestras inclinaciones destructivas, por lo que aún no se ha escrito su historia completa". Los humanos haríamos bien en asegurarnos de que cualquier máquina inteligente que diseñemos hoy y que pueda evolucionar hacia una máquina ultrainteligente lo haga de una manera que termine tratándonos bien. Como dice Eric Brynjolfsson: "El futuro no está predeterminado por las máquinas. Es creado por humanos".

## Sesgos en el desarrollo y aplicación de la IA y el Big Data

---

En la actualidad, se puede considerar que las tecnologías relacionadas con la inteligencia artificial se encuentran lo suficientemente implantadas como para influir sustancialmente en algunos aspectos de la vida diaria. Una de las áreas donde más ha crecido el uso de la inteligencia artificial en los últimos años es en el comercio electrónico, particularmente en los sistemas de recomendación y los asistentes que interactúan con los consumidores. Los agentes de inteligencia artificial, como los asistentes virtuales y los chatbots, ofrecen publicidad dirigida mediante la vinculación con bases de datos de empresas como es el caso de Amazon o Google y son capaces de responder de forma inmediata a las consultas de los usuarios sobre productos y servicios. Esto permite a las empresas obtener una respuesta rápida sin comprometer recursos humanos.

Sin embargo, la idea de que los algoritmos de inteligencia artificial están libres de sesgos es errónea debido a varias razones, algunas relacionadas con los sesgos existentes en los datos utilizados para aprender y otras a causa de los sesgos inherentes a los propios algoritmos.

Los sesgos en los datos pueden deberse a mediciones sesgadas, decisiones humanas sesgadas o informes erróneos. Es necesario considerar que los algoritmos de aprendizaje automático están diseñados básicamente para replicar estos sesgos. Otra razón proviene de los objetivos algorítmicos, que apuntan a minimizar los errores generales de predicción y, por lo tanto, benefician a los grupos mayoritarios sobre las minorías. Además, los sesgos pueden surgir de valores o información faltantes, como sesgo de muestra o selección, lo que da como resultado conjuntos de datos que no son representativos de la población objetivo.

Un ejemplo común de sesgo algorítmico está en el campo de la justicia penal, donde se demostró que un algoritmo empleado por el sistema de justicia penal estadounidense arrojaba como resultado que la probabilidad de reincidencia criminal de los ciudadanos afroamericanos era el doble que la de los blancos.

Puede consultarse al respecto los siguientes artículos:

- Angwin, J., J. Larson, S. Mattu, and L. Kirchner. 2016. «Machine Bias.» Disponible en: <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>
- Chouldechova, A. (2017). Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. In Big Data (Vol. 5, Issue 2, pp. 153–163). Mary Ann Liebert Inc.

Otro ejemplo de sesgo algorítmico se encontró en la atención médica, donde se demostró que un algoritmo de gestión ampliamente utilizado favorecía a los pacientes de raza blanca sobre los negros.

Para obtener más información al respecto se puede consultar el artículo:

- Obermeyer Z, Powers B, Vogeli C, Mullainathan S. Dissecting racial bias in an algorithm used to manage the health of populations. Science. 2019 Oct 25;366(6464):447-453. doi: 10.1126/science. aax2342. PMID: 31649194.

Así, la literatura ha señalado varias causas que pueden conducir a los sesgos en los programas basados en inteligencia artificial, que pueden tener su origen bien en los datos o bien en el propio algoritmo. Seguidamente se exponen las principales causas conocidas:

- **Sesgos debidos a los conjuntos de datos utilizados para el aprendizaje**, que se basan en mediciones de dispositivos sesgadas, decisiones humanas históricamente sesgadas, informes erróneos, bases de datos con grupos representados en proporciones distintas de las correctas, etc.
- **Sesgos causados por datos faltantes o bien sesgos en la selección de la muestra** que dan como resultados conjuntos de datos que no son representativos de la población objetivo.
- **Sesgos que surgen de objetivos algorítmicos que pretenden minimizar los errores de predicción agregados generales** y, por tanto, benefician a los grupos mayoritarios sobre las minorías. Este concepto está relacionado con la paradoja de Simpson, según la cual una tendencia observada en subgrupos puede comportarse de manera bastante diferente, incluso a la inversa, cuando estos subgrupos se agregan y analizan juntos.
- **Sesgos causados por variables relacionadas con los atributos sensibles**. Se entienden como atributos sensibles aquellos que diferencian a los grupos privilegiados y no privilegiados, como la raza, el género y la edad, y que, por lo general su uso no resulta legítimo para la toma de ciertas decisiones. Las variables relacionadas son variables no consideradas como sensibles pero que presentan una relación fuerte con dichas variables. En el caso de existir variables relacionadas dentro de un conjunto de datos, resultaría posible que el algoritmo de aprendizaje automático tomara decisiones implícitamente basadas en los atributos sensibles bajo la apariencia de usar atributos presuntamente legítimos pero que realmente son variables relacionadas.

Un ejemplo clásico de la paradoja de Simpson es el relativo a la supuesta discriminación por género que se producía en la Universidad de California en Berkeley (Estados Unidos) en el año 1973. Así, los datos de admisión mostraron que los hombres que presentaban solicitudes tenían más probabilidades de ser admitidos que las mujeres, y la diferencia era tan grande que era poco probable que se debiera a la casualidad.

Sin embargo, al analizar las admisiones departamento a departamento, se observaba que seis de los 85 departamentos tenían un sesgo significativo en contra de los hombres, mientras que cuatro lo tenían en contra de las mujeres. Es decir, realmente existía un pequeño sesgo favorable a las mujeres. De este análisis se concluyó que las mujeres tendían a postularse a departamentos más competitivos con tasas de admisión más bajas, mientras que los hombres tendían a hacerlo a departamentos menos competitivos con tasas de admisión más altas.

El Libro Blanco sobre la Inteligencia Artificial desarrollado por la Comisión Europea presenta los ejemplos que se muestran a continuación:

- «Puede suceder que el uso de determinados algoritmos de inteligencia artificial para predecir la reincidencia delictiva de lugar a prejuicios raciales o de género, y prevea una probabilidad de reincidencia distinta para hombres y mujeres o para nacionales y extranjeros. Fuente: Tolan S., Miron M., Gomez E. and Castillo C. «Why Machine Learning May Lead to Unfairness: Evidence from Risk Assessment for Juvenile Justice in Catalonia», Best Paper Award, International Conference on AI and Law, 2019.»
- «Algunos programas de inteligencia artificial de análisis facial muestran prejuicios raciales o de género, y presentan un bajo nivel de error a la hora de determinar el género de hombres de piel más clara, pero un elevado nivel de error al determinar el género de mujeres de piel más oscura. Fuente: Joy Buolamwini, Timnit Gebru; Proceedings of the 1<sup>st</sup> Conference on Fairness, Accountability and Transparency, PMLR 81:77-91, 2018.»

Por tanto, dado que para el entrenamiento y validación de los sistemas de inteligencia artificial es necesario hacer uso de conjuntos de datos y que las acciones y decisiones a las que pueden llevar dependen en gran medida de los conjuntos de datos que se hayan utilizado para entrenar los sistemas, deben adoptarse las medidas necesarias para garantizar que, en lo que se refiere a los datos utilizados para entrenar los sistemas de inteligencia artificial, se respeten los valores y normas de la UE, concretamente con relación a la seguridad y la legislación vigente para la protección de los derechos fundamentales.

Finalmente, cabe señalar que con gran frecuencia los desarrolladores otorgan a los dispositivos robóticos características antropomórficas, puesto que muchas personas desconfían o no les gusta hablar con máquinas. Teniendo en cuenta que múltiples agentes de inteligencia artificial pueden simular de manera convincente interacciones interpersonales, no es sorprendente que las personas interactúen con ellos como si fueran humanos. Los resultados obtenidos de la investigación sobre las relaciones interpersonales y las habidas entre el consumidor y las marcas, sugieren que las interacciones entre los humanos y los agentes inteligentes se guían por las mismas normas que rigen las relaciones interpersonales.

La investigación sobre los estereotipos sugiere que es más probable que las personas asocien a las mujeres con calidez y a los hombres con competencia (Cuddy, A. J. C., Fiske, S. T., & Glick, P. (2008). Warmth and Competence as Universal Dimensions of Social Perception: The Stereotype Content Model and the BIAS Map. In Advances in Experimental Social Psychology (pp. 61–149). Elsevier; Fiske, S. T., Cuddy, A. J. C., Glick, P., & Xu, J. (2002). A model of (often mixed) stereotype content: Competence and warmth respectively follow from perceived status and competition. In Journal of Personality and Social Psychology (Vol. 82, Issue 6, pp. 878–902). American Psychological Association APA). Los estudios también han demostrado que cuando las personas se comunican con dispositivos tecnológicos como computadoras, máquinas y entidades de inteligencia artificial, usan los mismos estilos de diálogo y métodos de procesamiento de información que utilizan en sus comunicaciones interpersonales humano a humano.

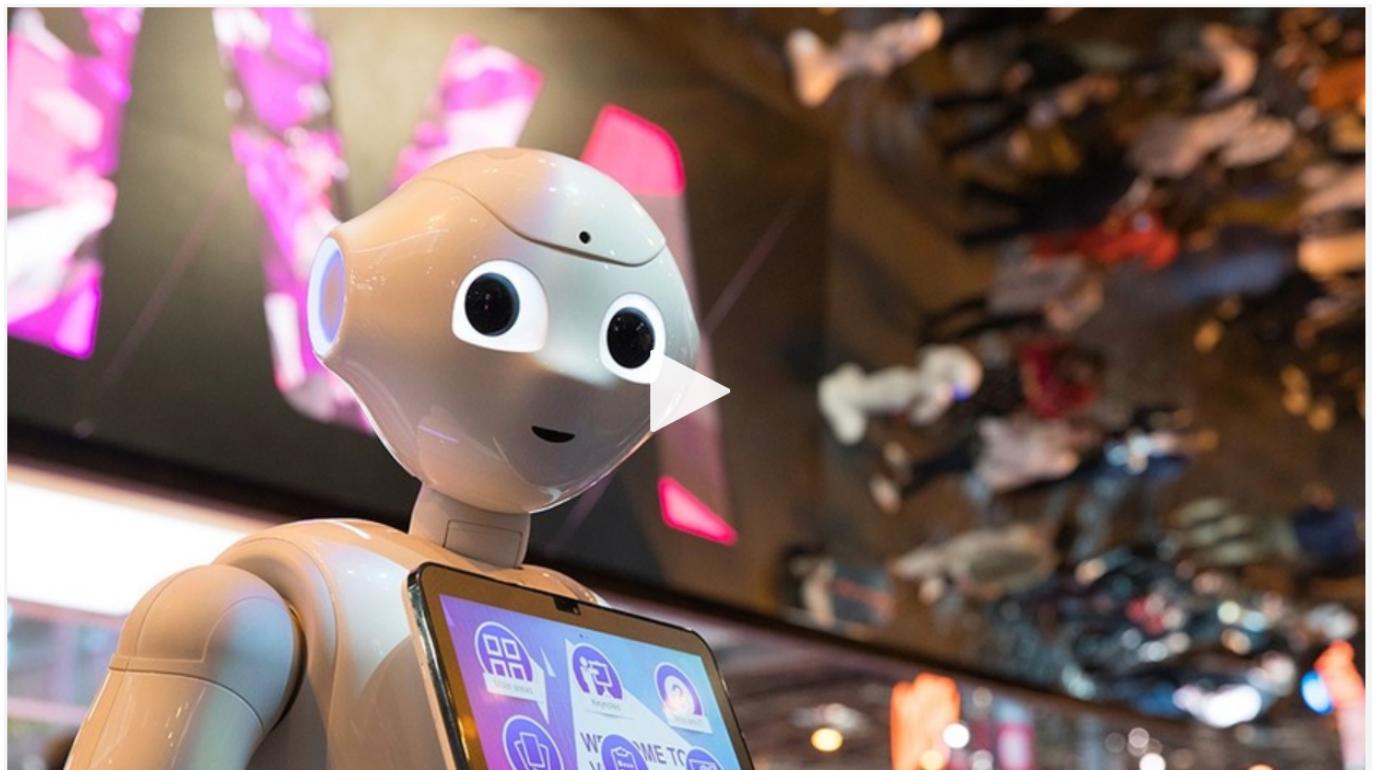
Por tanto, los estereotipos de género y en consecuencia sus sesgos, también se producen en las interacciones entre los humanos y los agentes inteligentes. Existen estudios que han demostrado que los usuarios no se comportan de igual manera cuando interactúan con un sistema de inteligencia artificial al que se le han otorgado rasgos de apariencia masculina que cuando

interaccionan con otro que tiene rasgos femeninos. Así, se ha manifestado que, por ejemplo, la capacidad de persuasión de las recomendaciones de los programas de inteligencia artificial varía según las personalidades percibidas por los usuarios. Existe un estudio en el que se llegó a la conclusión que, para los productos utilitarios, los participantes confiaron más en las recomendaciones de los agentes de inteligencia artificial con apariencia masculina que en las recomendaciones de los agentes de inteligencia artificial con apariencia femenina. Sin embargo, se produjo el caso contrario para los productos hedónicos.

## Regulación Europea sobre IA

En el marco de la Unión Europea se está desarrollando una comisión para examinar el impacto de la tecnología en general (y dentro de ella la Inteligencia Artificial). Se quiere definir una hoja de ruta europea a largo plazo que mantenga al ciudadano y su beneficio como el centro de cualquier desarrollo IA.

<https://multimedia.europarl.europa.eu/share/N01-PUB-200922-ARTI?lang=en&controls=on>



¿Debe haber un equilibrio entre tecnología y ley? ¿Quién y cómo debe definir límites a la IA?

Fuentes: - <https://www.europarl.europa.eu/topics/es/article/20230601STO93804/ley-de-ia-de-la-ue-primera-normativa-sobre-inteligencia-artificial> - [https://www.europarl.europa.eu/thinktank/es/document/EPRS\\_BRI\(2021\)698792](https://www.europarl.europa.eu/thinktank/es/document/EPRS_BRI(2021)698792) - [https://ec.europa.eu/commission/presscorner/detail/en/QANDA\\_21\\_1683](https://ec.europa.eu/commission/presscorner/detail/en/QANDA_21_1683)

⌚15 de julio de 2025

## 7.2 Actividades guiadas de la UD05

---

- [El algoritmo contra el crimen \(Documental\)](#)
- [Límites éticos para la Inteligencia Artificial \(Documental\)](#)
- [¡AI, AI, AI! Inteligencia artificial \(Documental\)](#)

 15 de julio de 2025

## 7.3 Actividades entregables de la UD05

---

### Relación de Legislación y Ética en IA con Series, Películas y Libros

Las series, películas y libros de ciencia ficción han sido cruciales para plantear y explorar los dilemas éticos y legales asociados con el avance de la inteligencia artificial y el big data. Aquí se presentan algunas obras adicionales que abordan estos temas y su relevancia para nuestra tertulia.

#### **Black Mirror**

- **Temas Abordados:**

- **Privacidad y Datos Personales:** Episodios como "Nosedive" y "The Entire History of You" exploran cómo la recopilación y el uso de datos personales pueden afectar la privacidad y la autonomía individual.
- **Responsabilidad y Rendición de Cuentas:** "White Bear" y "Hated in the Nation" plantean preguntas sobre la responsabilidad en una sociedad altamente vigilada y conectada.
- **Relevancia:** Black Mirror nos enfrenta a escenarios distópicos donde la tecnología se utiliza de manera que vulnera derechos fundamentales, invitándonos a reflexionar sobre los límites éticos y legales que deberíamos establecer.

#### **Westworld**

- **Temas Abordados:**

- **Autonomía y Control Humano:** La serie explora el concepto de conciencia en las IA y la lucha por la autonomía de los anfitriones.
- **Responsabilidad y Ética:** Cuestiona la moralidad de crear seres conscientes para el entretenimiento humano y las implicaciones éticas de sus acciones.
- **Relevancia:** Westworld nos lleva a considerar hasta qué punto las IA pueden ser consideradas entidades con derechos y cómo deberíamos legislar su existencia y tratamiento.

#### **Ex Machina**

- **Temas Abordados:**

- **Transparencia y Explicabilidad:** La película cuestiona la transparencia en la creación y el funcionamiento de las IA, así como las intenciones detrás de su desarrollo.
- **Equidad y No Discriminación:** Aborda los sesgos inherentes en la creación de IA y las dinámicas de poder entre creadores y criaturas.
- **Relevancia:** Ex Machina nos invita a reflexionar sobre la necesidad de transparencia y ética en la investigación y desarrollo de IA, y sobre cómo los sesgos pueden influir en su comportamiento.

#### **Her**

- **Temas Abordados:**

- **Privacidad y Relaciones Humanas:** Explora la privacidad en las interacciones con IA y cómo estas relaciones pueden impactar la vida personal.
- **Responsabilidad y Autonomía:** Cuestiona la naturaleza de la relación entre humanos y IA, y la responsabilidad emocional y ética hacia las IA avanzadas.
- **Relevancia:** Her destaca la necesidad de considerar el impacto emocional y social de la IA en nuestras vidas, además de los desafíos éticos que surgen de las relaciones cada vez más personales con la tecnología.

#### **The Matrix**

- **Temas Abordados:**

- **Autonomía y Control:** Plantea la cuestión del control y la manipulación de la realidad por parte de sistemas de IA.
- **Impacto Social y Económico:** Explora un mundo donde la IA ha tomado el control y los humanos son relegados a una existencia servil.
- **Relevancia:** The Matrix nos lleva a cuestionar la relación de poder entre humanos y máquinas, y las implicaciones éticas y sociales de un mundo dominado por IA.

### Person of Interest

- **Temas Abordados:**

- **Vigilancia y Privacidad:** La serie aborda el uso de la IA para la vigilancia masiva y la prevención de delitos, cuestionando hasta qué punto es ético sacrificar la privacidad por la seguridad.
- **Responsabilidad y Ética:** Explora las consecuencias de un sistema autónomo tomando decisiones críticas y el papel de los humanos en supervisarlo.
- **Relevancia:** Person of Interest nos hace reflexionar sobre el balance entre seguridad y privacidad, y sobre los límites éticos y legales de la vigilancia tecnológica.

### Minority Report

- **Temas Abordados:**

- **Predicción y Pre-crimen:** La película se centra en la capacidad de predecir delitos antes de que ocurran y las implicaciones éticas y legales de castigar a personas por crímenes que aún no han cometido.
- **Libertad y Determinismo:** Cuestiona la libertad individual frente a la capacidad de la tecnología para determinar comportamientos futuros.
- **Relevancia:** Minority Report plantea preguntas cruciales sobre la predicción basada en big data y la justicia, y cómo podemos regular tales tecnologías para proteger los derechos individuales.

### 1984 de George Orwell

- **Temas Abordados:**

- **Vigilancia y Control Totalitario:** El libro describe una sociedad donde el gobierno utiliza la tecnología para la vigilancia y el control total de los ciudadanos.
- **Manipulación y Propaganda:** Explora cómo la información puede ser manipulada para mantener el poder.
- **Relevancia:** 1984 es una advertencia sobre los peligros del control totalitario y la vigilancia masiva, relevando la importancia de legislar para proteger la libertad y la privacidad.

### El Hombre Bicentenario

- **Temas Abordados:**

- **Derechos de las IA:** La película aborda la evolución de un robot hacia la humanidad y la lucha por el reconocimiento de sus derechos como ser consciente.
- **Identidad y Autonomía:** Explora la búsqueda de identidad y autonomía de las IA, cuestionando qué significa ser humano.
- **Relevancia:** El Hombre Bicentenario nos hace considerar hasta qué punto las IA pueden y deben tener derechos, y cómo debemos adaptar nuestras leyes para reconocer y proteger la autonomía de seres no humanos.

### Cassandra (Netflix)

- **Temas Abordados:**

- **Asistentes con IA:** ¿Qué pasaría si uno de los primeros Asistentes virtuales domóticos se hubiese diseñado y creado en los años 60?
- Sinopsis: La casa inteligente más antigua de Alemania despierta a su ayudante IA **Cassandra** tras décadas 'dormida' cuando una nueva familia se muda a ella.
- **Relevancia:** Es la historia de una IA adelantada a su tiempo, controlada por un ordenador más grande que tu propia casa y con muchos problemas de diseño. Problemas que permiten **profundizar en cuestiones ética y moralmente cuestionables, relacionadas con el uso de la IA en el día a día.**

### Years and Years (BBC/HBO, 2019)

- **Temas Abordados:**

- **Vigilancia Masiva y Autoritarismo Digital:** Una política populista usa IA para controlar a la población mediante reconocimiento facial y crédito social.
- **Impacto Social de la Tecnología:** Muestra cómo la IA profundiza las brechas económicas y políticas.
- **Relevancia:** Un paralelo escalofriante con sistemas actuales como el de China, ideal para debatir regulaciones.

15 de julio de 2025



## 8. Sobre mí...

### 8.1 David Martínez Peña

### 8.2 Contacto:

- ✉ d.martinezpena@edu.gva.es
- ▶ Youtube
- 💼 LinkedIn
- 🐙 GitHub



 Profesor de Secundaria, especialidad de Informática.

 Actualmente con destino en IES Eduardo Primo Marqués de Carlet

 Modelos de Inteligencia Artificial © 2025 by David Martínez is licensed under CC BY-NC-SA 4.0



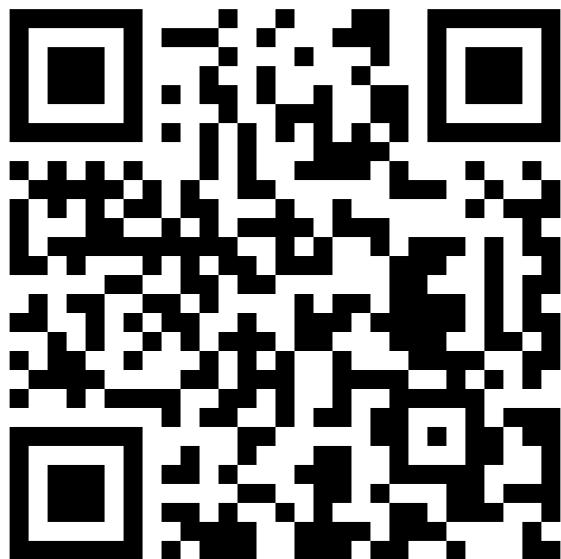
⌚ 14 de julio de 2025

## 9. Fuentes de información

---

- Wikipedia
- GhatGPT
- Modelos de Inteligencia Artificial (Ed. Marcombo)
- <https://iep.utm.edu/artificial-intelligence/>
- Materiales MIA curso MEC-20230524
- <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>
- <https://rapidapi.com/blog/top-facial-recognition-apis/>
- <https://www.sciencedirect.com/science/article/pii/S0212656720301463>
- Models d'IA de Carles Gonzalez està sotmés a la llicència: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International 
- Introducción al Procesamiento del Lenguaje Natural (PLN) Ferran Pla 2017
- Artificial Intelligence a modern approach 4<sup>th</sup> edition (Peter Norvig)
- <https://www.jetbrains.com>
- ChatGPT
- DeepSeek

 16 de julio de 2025



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

David Martínez Peña

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