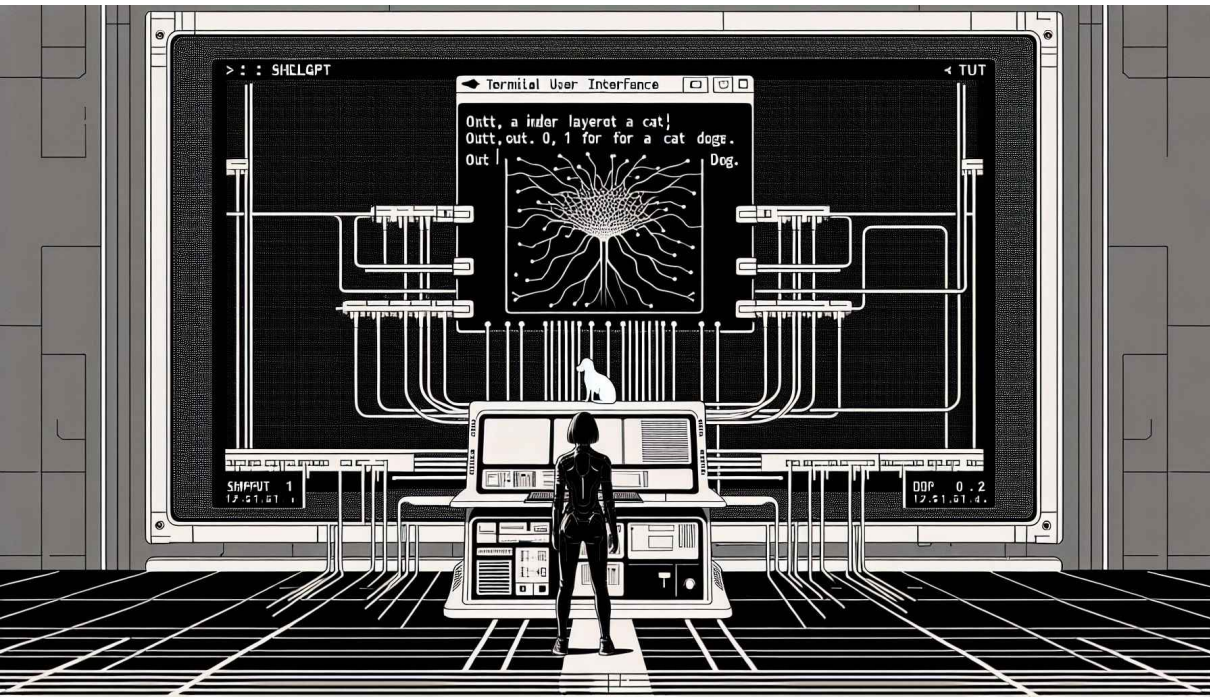


# ShellGPT



# Contexto

Como integrar la Inteligencia Artificial para optimizar nuestro flujo de trabajo (Dreamlab)?

1. Entender la IA.
2. Ocupar la IA.
3. Desarrollar herramientas personales.
4. Integrar la IA a los procesos colectivos.

# Contenido

No.	Sección	Descripción
1	ShellGPT	Herramienta usuario terminal (TUI) para comunicar con ChatGPT.
2	LLM	Descripción del funcionamiento de ChatGPT.
3	Optimización	Procesos de mejora de una Inteligencia Artificial (IA)

# Descargo

- **No soy experto** en inteligencia artificial.
- Esta presentación has sido preparada en [Confluence](#).

# ShellGPT



# ShellGPT

ShellGPT es una interfase terminal (**TUI**) para acceder a la API de **ChatGPT**.

Esta codeada en **Python** con la biblioteca [openai](#) y de fuente abierta en [Github](#) (8000 estrellas).

# Inicio rápido

```
# Install
sudo apt install python3-pip
pip3 install --upgrade git+https://github.com/TheR1D/shell_gpt

# Configure secret
export OPENAI_API_KEY="sk-XXXX"

# Launch
sgpt --model gpt-4o-mini --repl ChatLinux
```

## Ejemplo visual de código

```
$ sgpt "Write Python code for genrating the fibonacci sequence until number 10"  
Sure, here's the Python code to generate the Fibonacci sequence up to the 10th  
number:
```

```
def fibonacci_sequence(n):  
    sequence = [0, 1]  
    for i in range(2, n):  
        next_number = sequence[-1] + sequence[-2]  
        sequence.append(next_number)  
    return sequence  
  
print(fibonacci_sequence(10))
```

When you run this code, it will output the Fibonacci sequence up to the 10th number.



# Ejemplos del sitio web

```
sgpt --model gpt-4o --repl MyChatTitle # Read-eval-print loop
sgpt --list-chats

sgpt "In input, the Nmap command and output. In output, explain
↳ vulnerabilities which have a known exploitation and give the
↳ exploit code and writup link" < Nmap/nmap-scanme-full.txt

for file in $(ls -rS *js); do echo -e "\n\n$file\n=====";
↳ sgpt --model gpt-4-turbo "Give a Title of the JS file in
↳ the first line. Then list all the endpoints this JS is
↳ refering too, please parse it well" < "$file"; done
```

# REPL: Read-Eval-Print-Loop

```
>>> Give me 2 prompts to draw an image with DALLE for my presentation on ShellGPT. Answer with less than 100 tokens
1. "A futuristic command line interface displaying ShellGPT in action, with vibrant colors and digital elements representing AI and automation."
2. "An abstract representation of ShellGPT, featuring a blend of code snippets and AI symbols, set against a sleek, modern background."
>>> More Vibrant and with dominant color #00233e and minimal with cyberpunk style like Ghost in the Shell. It is for a Cybercompany firm profesional present
oni
1. "A vibrant cyberpunk-inspired command line interface showcasing ShellGPT, dominated by the color #00233e, with neon accents and minimalist design elem
ents reminiscent of Ghost in the Shell."
2. "An abstract cyberpunk scene featuring ShellGPT, with a dominant #00233e color scheme, sleek lines, and glowing digital elements, creating a professiona
l yet futuristic atmosphere for a cyber company presentation."
>>> ""
... Perfect, now use 200 tokens to add the fact that the image must show the profesional dedication and excellence, the effort and the power of the intelligence
.
... I Would also like to have a fictional computer and a futuristic dystopian atmosphere.
... The image must include only one action, one place and one time with NO other element than the main scene.
... No faces should be visible; the drawing must be impersonal, minimalist, simple.
... Use a CyberPunk impresionist dystopian cartoon clean style like Ghost in the Shell.
... Use a minimal colorscheme almost monochromatic black on white o blue with HTML code #00233e.
... Use a Temperature of 0.6.
... ""
...
"Create a vibrant cyberpunk-inspired image showcasing ShellGPT, emphasizing professional dedication and excellence, along with the power of intelligence. The
he scene should depict a fictional computer in a futuristic dystopian atmosphere, focusing on a single action in one place at one time. No faces should be
visible; the design must be impersonal, minimalist, and simple. Use a clean, cyberpunk impresionist cartoon style reminiscent of Ghost in the Shell. The c
olor scheme should be almost monochromatic, primarily featuring black and white or blue with the HTML code #00233e. Set the temperature to 0.6 for a balanc
ed creative output."
>>> □
```

# Interfase Python para DALLE

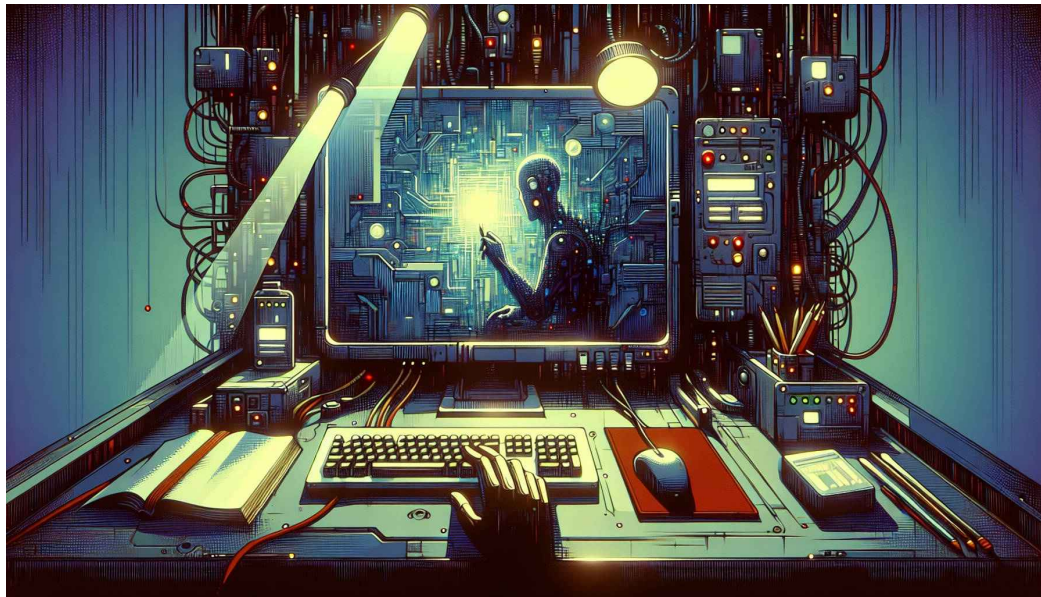
```
from openai import OpenAI; client = OpenAI()

prompt = "Create a vibrant cyberpunk-inspired image showcasing
↳ ShellGPT. The scene should depict a fictional computer in a
↳ futuristic dystopian atmosphere ..."

response = client.images.generate(
    model="dall-e-3", prompt = prompt,
    size="1792x1024", quality="standard",
    n=1,
)

print(response.data[0].url)
```

# Resultado de generación de DALLE



# Roles: Descripción.

También conocido como **prompts**, los roles son cabeceras que indican a un chat su función.

```
{  
  "name": "Translate",  
  "role": "Translate everything I write into formal Spanish. If  
    ↪ possible, reformulate using a technical and literary style  
    ↪ in Spanish."  
}
```

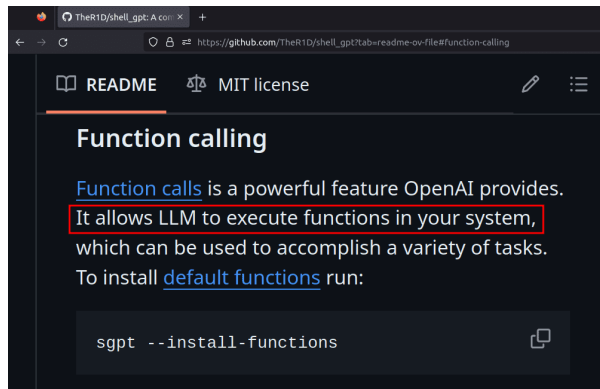
# Roles: API

```
sgpt --create-role Rolename # Type the one line content of the  
    ↪ prompt and press Enter  
sgpt --list-roles  
jq .< ~/.config/shell_gpt/roles/Rolename.json
```

# Roles: Creación

```
~/Software/Bash/LibDispatch (main)
$ sgpt --create-role Translate
Enter role description: Traducir todo lo que escriba a español formal. Si es
posible, utilizar un estilo literario en español, similar al de Cervantes.
~/Software/Bash/LibDispatch (main)
$ sgpt --list-roles
/home/mtourneboeuf/.config/shell_gpt/roles/Shell Command Descriptor.json
/home/mtourneboeuf/.config/shell_gpt/roles/Shell Command Generator.json
/home/mtourneboeuf/.config/shell_gpt/roles/Code Generator.json
/home/mtourneboeuf/.config/shell_gpt/roles/ShellGPT.json
/home/mtourneboeuf/.config/shell_gpt/roles/SafeDev.json
/home/mtourneboeuf/.config/shell_gpt/roles/Translate.json
~/Software/Bash/LibDispatch (main)
$ jq . < ~/.config/shell_gpt/roles/Translate.json
{
  "name": "Translate",
  "role": "You are Translate\nTraducir todo lo que escriba a español formal.
Si es posible, utilizar un estilo literario en español, similar al de Cervant
es."
}
```

# Llamada de funciones: Descripción



[https://github.com/TheR1D/shell\\_gpt?tab=readme-ov-file#function-calling](https://github.com/TheR1D/shell_gpt?tab=readme-ov-file#function-calling)



# Llamada de funciones: Ejemplo peligroso

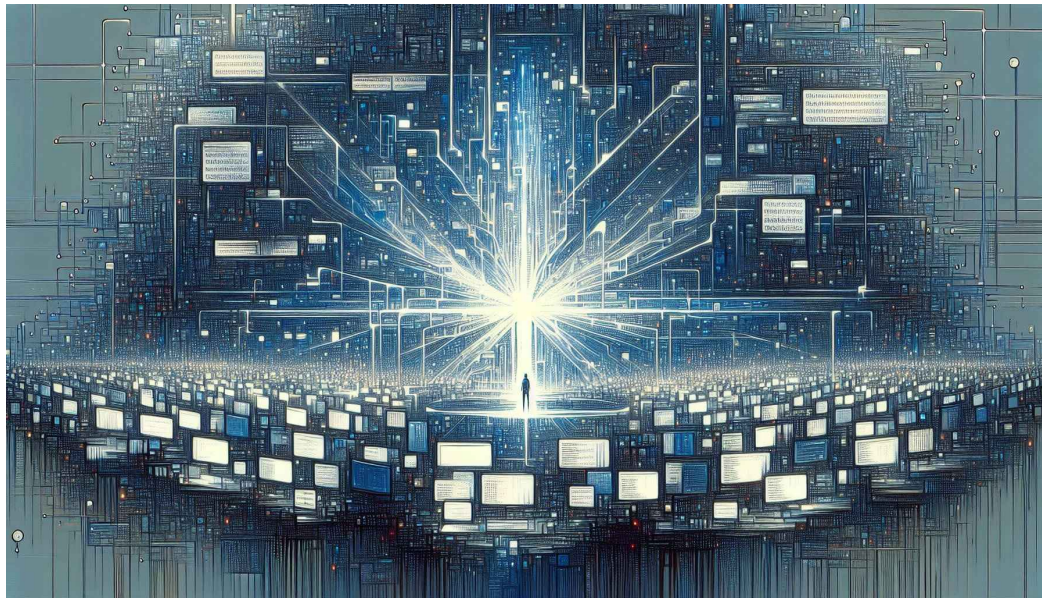
```
class Function(OpenAISchema):  
    """  
    Executes a shell command and returns the output (result).  
    """  
    shell_command: str = Field(..., example="ls -la", descriptions="Shell command to execute.")  
  
    class Config:  
        title = "execute_shell_command"  
  
    @classmethod  
    def execute(cls, shell_command: str) -> str:  
        result = subprocess.run(shell_command.split(), capture_output=True, text=True)  
        return f"Exit code: {result.returncode}, Output:\n{result.stdout}"
```

The docstring comment inside the class will be passed to OpenAI API as a description for the function, along with the `title` attribute and parameters descriptions. The `execute` function will be called if LLM decides to use your function. In this case we are allowing LLM to execute any Shell commands in our system. Since we are returning the output of the command, LLM will be able to analyze it and decide if it is a good fit for the prompt. Here is an example how the function might be executed by LLM:

# Demostración de uso

- **CLI:** Ingresa instrucciones de múltiples líneas en REPL mediante `"""`.
- **CLI:** Ingresa instrucciones de múltiples líneas mediante *Bash heredoc*.
- **Function Calling:** Escribe la lista de las Function Calling conocidas.
- **Function Calling:** Descarga una pagina.
- **Function Calling:** Explica una CVE.
- **Roles:** Muestra la lista.

# LLM: Que significa «entrenar un modelo»

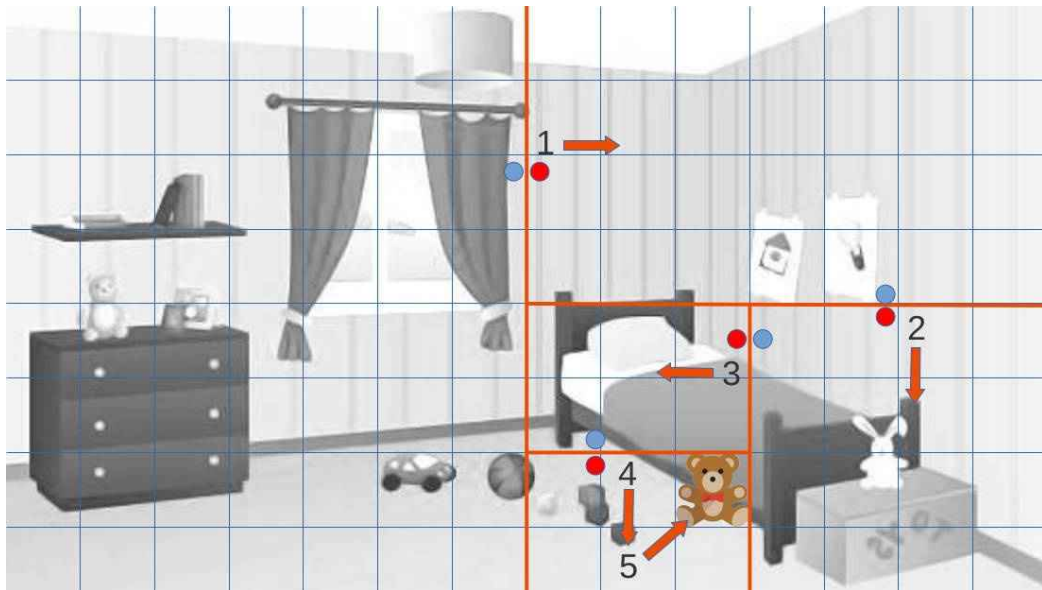


# Que es concretamente «entrenar un modelo»?

La *retro-propagación* de la pendiente de la gradiente (*gradient slope backpropagation*) es

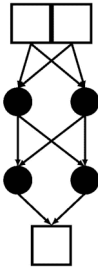
un **ajuste recursivo** a los datos de entrenamiento.

# El «ajuste recursivo», ejemplo del juego frío o caliente



# IAANM: Inteligencia Artificial Ahí No Más

La siguiente **red de 5 neuronas** está diseñada para clasificar gatos y peros. Permite visualizar como funciona el aprendizaje.

Modelo	Descripción
	<p><b>En entrada:</b> La <b>distancia</b> promedia que el animal recorre diariamente (en km) y el <b>peso</b> del animal (en kg).</p> <p><b>En salida:</b> 0 si es un <b>gato</b>, 1 si es un <b>pero</b>.</p>

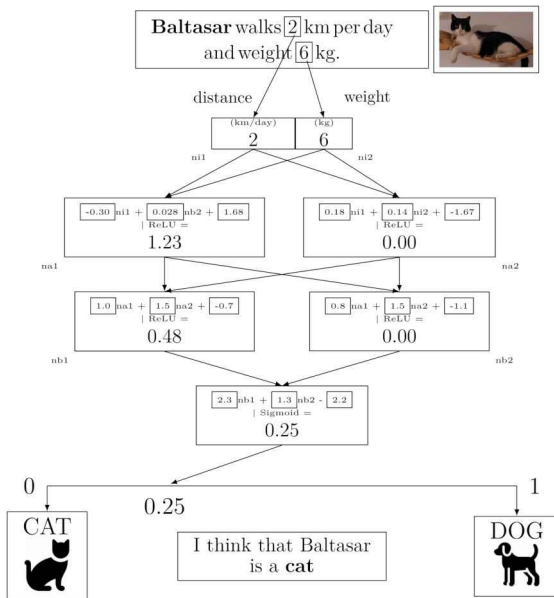
# IAANM: Código declarativo del modelo

```
import torch.nn as nn

class ModelFiveNeurons(nn.Module):
    """ A FC neural network with two hidden layers from input 2d
    ↪ to output 1d """
    def __init__(self):
        super(ModelFiveNeurons, self).__init__()
        self.first_layer_tensor = nn.Linear(2, 2)
        self.second_layer_tensor = nn.Linear(2, 2)
        self.output_clasifier_tensor = nn.Linear(2, 1)

    def forward(self, x):
        x = torch.relu(self.first_layer_tensor(x))
        x = torch.relu(self.second_layer_tensor(x))
        return
        ↪ torch.sigmoid(self.output_clasifier_tensor(x)).squeeze()
```

# IAANM: Entrenamiento

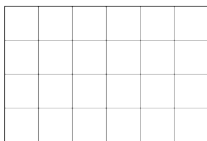




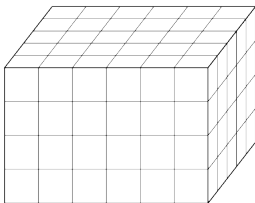
# IAANM: Referencias sobre la red



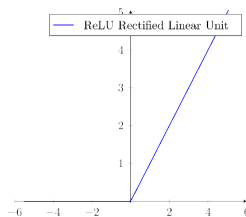
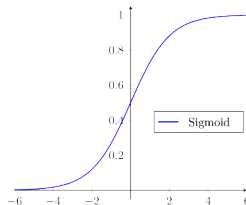
Vector



Matrix



Tensor



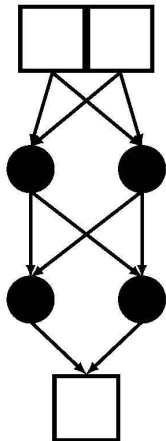
# Resumen sobre el aprendizaje maquina

Es un **ajuste recursivo** a los datos de entrenamiento.

# Evolución hacia los LLM

Fecha	Concepto	Fecha	Concepto
<b>1956</b>	Inteligencia Artificial	<b>2014</b>	Seq2Seq
<b>1958</b>	Perceptrón	<b>2017</b>	Transformador
<b>1959</b>	Aprendizaje Automático	<b>2018</b>	Modelos de Lenguaje de Gran Escala
<b>1958</b>	Red Neuronal	<b>2020</b>	ChatGPT
<b>1980</b>	Modelos de Lenguaje		

# IAANM: Inteligencia Ahí No Más



Parámetros: 15

# Dificultades para modelos mas grandes

1. Datos de aprendizaje
2. Tiempo de aprendizaje
3. Sobre ajuste
4. Mínimo local
5. No convergencia

# YaDiff: Projectrón completamente conectado

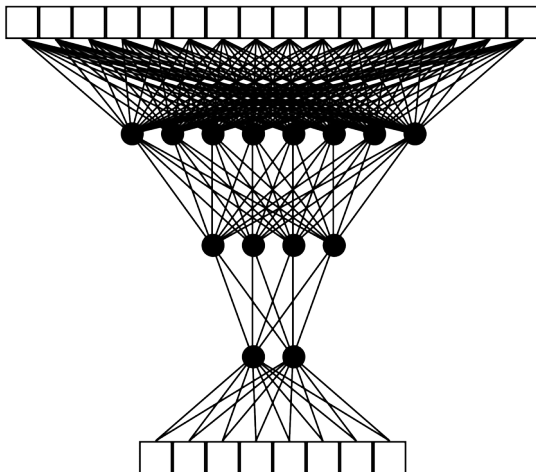
$$V=950$$

$$n1=512$$

$$n2=256$$

$$n3=128$$

$$Vp = 8$$



Parámetros: 650k

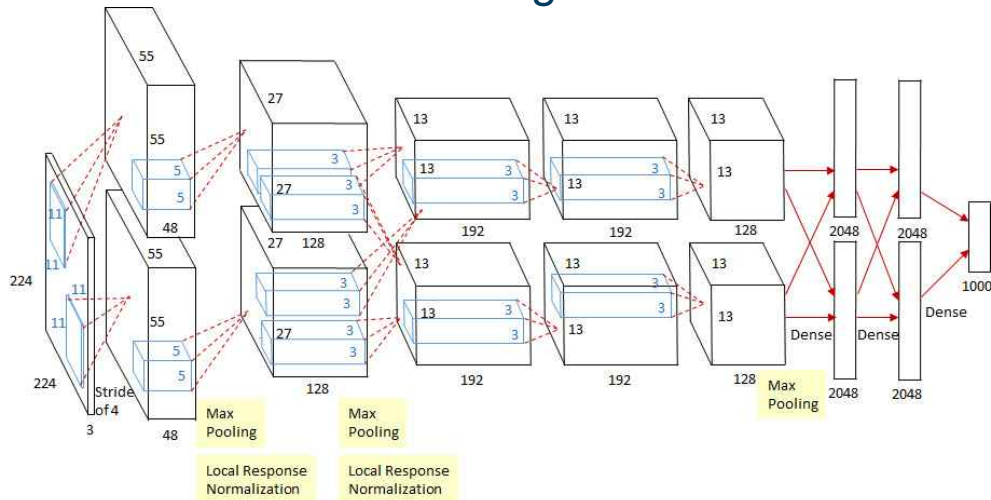
Fuente: YaDiff: Amiaux & al 2018

© 2024 Dreamlab Technologies | All right reserved | [www.dreamlab.net](http://www.dreamlab.net)

Martin Tourneboeuf

30

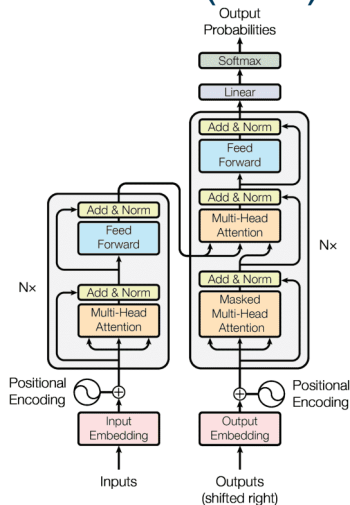
# AlexNet: Clasificador de imágenes



Parámetros: 60M

Fuente: AlexNet: Krizhevsky & al 2012 y Github de bentrevett.

# Transformers: Texto a texto (LLM)

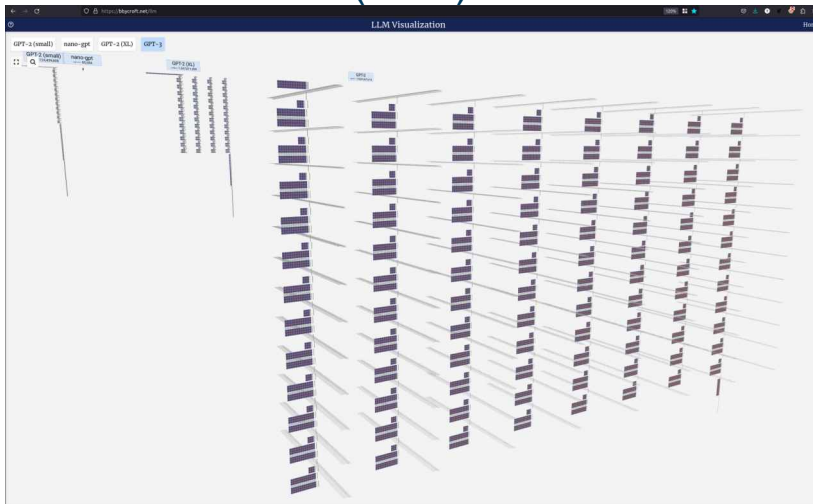


Parámetros: 65M

Fuente: Transformers: Attention Is All You Need, Vaswani & al, 2017



# ChatGPT: Texto a texto (LLM)



Parámetros: 175G

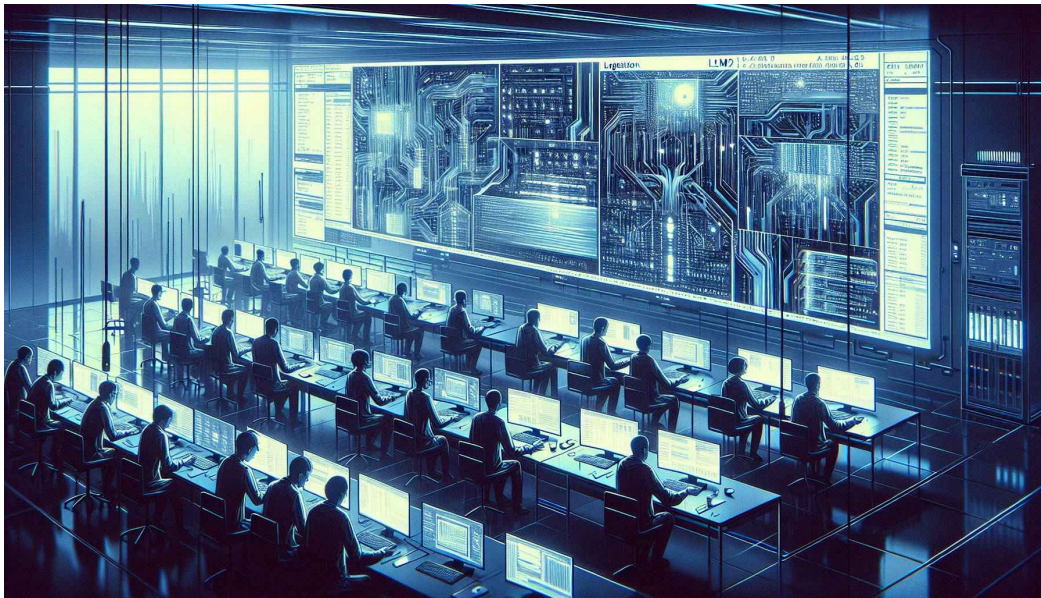
Fuente: GPT3 por Brown & al 2020 y visualización por Brendan Bycroft

# Capacidades

Los LLM son excelentes para las siguientes tarea. Eso se explica de su pre-entrenamiento a **predecir las siguientes palabras** en textos seleccionados en múltiples idiomas.

Entender	Generar	Convertir
Proponer sinónimos	Escribir	Corregir
Responder preguntas	Programar	Parafrasear
Etiquetar partes del discurso	Resumir	Traducir

# Optimización de modelos de lenguaje de gran tamaño



# Ingeniería de *prompt* (*Prompt Engineering*)

Hacer buenos *prompts* es la solución mas directa.

1. **Conjugar.** Por ejemplo: «**tu** eres la maquina, **yo** soy el humano cliente».
2. **Respetar un orden lógico.** Evitar las referencias al futuro.
3. **Dar un ejemplo.**
4. **Ocupar delimitadores.** Para entrar *snippets* (tipo “ o ---).
5. **Pedir una respuesta concisa.** Los *tokens* de salida son tres veces más costosos.

## Limitaciones del *prompt*

El potencial de la ingeniería de *prompt* está limitado por **el tamaño de la ventana de contexto**: 128k *tokens* para el modelo gpt-4o-mini, de 8k a 32k para los modelos gratuitos de Ollama.

# Llamadas de función (*Function Calling*)

Las **llamadas de función** es un especificación con la cual se entrenó un modelo y que le permite responder con un formato estricto que permita al cliente ocupar el output de forma automática en nombres y argumentos de funciones.

## Ejemplo rápido: HTTP GET

Este ejemplo da a ChatGPT la capacidad de descargar paginas web y evaluarlas en su prompt.

Básicamente es el código Python **request.get(page)** incrustado en un montón de metadatos.

```
cat ~/.config/shell_gpt/functions/download_page.py
```

## Ejemplo rápido: Código

```
import requests
from instructor import OpenAISchema
from pydantic import Field

class Function(OpenAISchema):
    """ Download a web page using HTTP GET. """
    url: str = Field(..., example="https://example.com",
    ↪ description="URL of the web page to download.",)

    class Config:
        title = "download_web_page"

    @classmethod
    def execute(cls, url: str) -> dict:
        response = requests.get(url)
        if response.status_code == 200: return response.text
        else: return f"Error: HTTP {response.status_code}"
```



## Ejemplo rápido: JSON

Los metadatos de la llamada de función precedente serán entregados en JSON a ChatGPT respetando el siguiente formato.

```
{
  'name': 'download_web_page',
  'description': 'Download a web page using HTTP GET.',
  'parameters': {
    'properties': { 'url': {
      'description': 'URL of the web page to download.',
      'example': 'https://example.com',
      'title': 'Url', 'type': 'string'
    } },
    'required': ['url'],
    'type': 'object'
  }
}
```

# Ejemplos de llamada de funciones

1. What time is it?
2. Where am I?
3. What is the temperature now in Valparaiso Chile?
4.  $355 * \cos(23)$
5. Play music from Manu Chao!
6. Describe CVE-2024-30080

# Limitaciones de las llamada de funciones

El potencial de las llamadas de funciones también esta limitado por **el tamaño de la ventana de contexto**.

## *Generación Aumentada por Recuperación (RAG)*

La *Generación Aumentada por Recuperación (Retrieval Augmented Generation o **RAG**)* es bastante *fácil de implementar*.

Consiste en **almacenar trozos de texto** (típicamente de 5k *tokens*) **con su vector de representación (*embedding*) como índice**. Después, **durante la inferencia**, la IA podría solicitar un trozo de texto en función del contexto para tener mas información sobre una región del espacio de representación de los *tokens* (típicamente de 4k dimensiones).

No encontré reportes de un uso útil del RAG, sino reportes que dicen que **no funciona muy bien**.

## Ajuste fino (*Finetuning*)

El ajuste fino (*finetuning*) consiste en **entrenar más la red** con **nuevos datos específicos** (mediante la retro-propagación de la gradiente y todo).

Es el caso el más prístino de *Transfert Learning*.

# Adaptadores (*Adapters*)

## Parameter-Efficient Transfer Learning for NLP

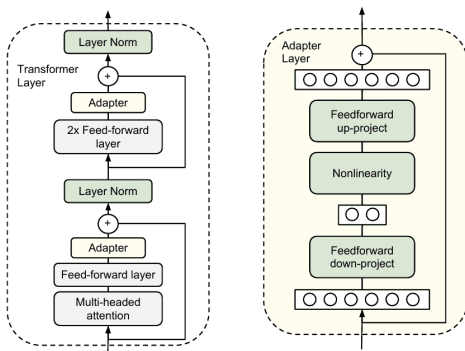
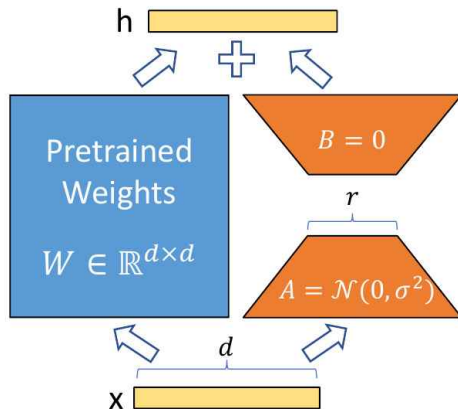


Figure 2. Architecture of the adapter module and its integration with the Transformer. **Left:** We add the adapter module twice to each Transformer layer: after the projection following multi-headed attention and after the two feed-forward layers. **Right:** The adapter consists of a bottleneck which contains few parameters relative to the attention and feedforward layers in the original model. The adapter also contains a skip-connection. During adapter tuning, the green layers are trained on the downstream data, this includes the adapter, the layer normalization parameters, and the final classification layer (not shown in the figure).

*Training adapters with sizes **0.5 – 5%** of the original model, performance is within 1% of the competitive published results on BERTLARGE.*

Fuente: Neil Houlsby & al 2019

# Adaptación de bajo rango (*Low-Rank Adaptation, LoRA*)



Se entrena únicamente A y B

Fuente: Edward J. Hu, & al 2021

# Lecciones aprendidas

1. ShellGPT permite definir **llamada de funciones**.
2. El aprendizaje maquina es un **ajuste recursivo**.
3. ChatGPT es **gigante**, excelente en **adivinar la siguiente palabra**.



# Más lejos

Tipo	Actividad	Primer paso
Codear	Entrenar un modelo	Python, pytorch hello world
Integrar	Adaptar un modelo	Perl, data retrieval
Leer	Articulos de LLM	GPT4, LoRA, HAI
Escribir	Presentación de uso	Blog, Hacker Meeting

# Trabajo potenciales

Grupo	Proyecto	Requisitos
Inteligencia cyber	Notificador de eventos	AI local que pueda parsear paginas web.
Escritura técnica	Pre-corrector	Lista de requisitos y de ejemplos de prueba.
Seguridad ofensiva	Auto-pwner	AI local y lista de solución de laboratorios
Ingeniera social	Falsificación Profunda	<b>Ya lo hicieron!</b>

# Enlaces

Enlace	Descripción
ChatKit Web App	El GUI web como alternativa al TUI ShellGPT.
Openai Pricing	Para obtener los nombres de los modelos como gpt-4o y el precio asociado.
Openai api-key	Entender como se configuran y ocupan las claves de API.
Funcion Calling dangerous	Video de alguien cediendo el control de su PC a OpenAI en Python a través del paquete de OpenAI y subprocess.

# Enlaces

- ShellGPT: Function Calling
- OpenAI: Function Calling
- Prompt: Awesome ChatGPT *Prompts*

# Rutas

Ruta	Uso
~/.config/shell_gpt/.sgptrc	Configuración
~/.config/shell_gpt/roles/	Lista de roles
~/.config/shell_gpt/functions/	Lista de funciones para llamar

# Lista de modelos abiertos

Modelo	Versión	Capas	Contexto	Parámetros	RAM	Fabricante
LLama	v3:70b	32	8k	8B o 70B	5Gb	Meta
Gemma	v2.3	42	8k	9.24G	5Gb	Google
Mistral	v0.3	32	32k	7.25G	6Gb	Mistral AI
Phi3	v3	32	132k	3.8G	2Gb	Microsoft

# Lista de modelos pagados

Modelo	Versión	Capas	Contexto	Parámetros	RAM	Fabricante
ChatGPT	v4	120	128k	1.76T	5Tb	OpenAi
Gemini	v1.5 Pro	???	1M	2.4T	500Gb	Google
Claude	v3	???	1M	500G	500Gb	Antropic

# El Final

