# TL06 LLMs

## Índice

# 1 Introducción

**Documentación de transformers:**

- **Get started:** TL02
- **Clases base:** TL03
- **Inferencia:**
  - **API Pipeline:** TL03
  - **LLMs:** veremos 3 apartados
    - **Generación de texto:** sección 2
    - **Estrategias de generación:** sección 3
    - **Ingeniería de prompts:** sección 4

**Otras librerías y recursos:**

- **pytorch** TL01
- **datasets:** TL04
- **evaluate:** TL04
- Benchmarking: sección 5

**Objetivo:** introducir generación de texto, estrategias de generación, prompting y benchmarking

# 2 Generación de texto

**Text generation:** con generate()

**Modelo ejemplo:** `Qwen/Qwen3-0.6B`

In [ ]:
```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_name = "Qwen/Qwen3-0.6B"
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype="auto", device_map="auto")
tokenizer = AutoTokenizer.from_pretrained(model_name) # padding_side="left"
```

In [ ]:
```
model_inputs = tokenizer(["Lista de colores: rojo, azul"], return_tensors="pt").to(model.device)
model_inputs
```

Out[ ]:
```
{'input_ids': tensor([[43617,   409,  1375,  4589,    25,   926,  7305,    11, 12376,   360]],
       device='cuda:0'), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], device='cuda:0')}
```

In [ ]:
```
generated_ids = model.generate(**model_inputs)
generated_ids
```

Out[ ]:
```
tensor([[43617,   409,  1375,  4589,    25,   926,  7305,    11, 12376,   360,
            11, 73161,    11, 89547,    11,   308, 21320,  5580,    11,   296,
          1118,  3165,    11,  1079,   277, 21782,    11,   926,  7305,    11]],
       device='cuda:0')
```

In [ ]:
```
tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

Out[ ]:
```
'Lista de colores: rojo, azul, verde, blanco, naranja, marrón, amarillo, rojo,'
```

**Nota:** observa que la coma produce el token 11, azul los tokens 12376 y 360, verde 73161, etc.

### Configuración:

In [ ]:
```
model.generation_config
```

Out[ ]:
```
GenerationConfig {
  "bos_token_id": 151643,
  "do_sample": true,
  "eos_token_id": [
    151645,
    151643
  ],
  "pad_token_id": 151643,
  "temperature": 0.6,
  "top_k": 20,
  "top_p": 0.95
}
```

### Parámetros:

- `generate()`: `inputs`, `generation_config`, `logits_processor`, ..., `kwargs`
- `generation_config()`: `max_length`, `max_new_tokens`, `temperature`, etc.
- `kwargs` de `generate()` incluye parámetros ad hoc de `generation_config` y específicos del modelo

In [ ]:
```
generated_ids = model.generate(**model_inputs, max_length=36)
tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

Out[ ]:
```
'Lista de colores: rojo, azul, verde, blanco, negro, marrón, lila, amarillo, rojo, verde, marrón'
```

In [ ]:
```
generated_ids = model.generate(**model_inputs, temperature=1.0)
tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

Out[ ]:
```
'Lista de colores: rojo, azul, negro, blanco, gris, blanco, azul, negro, gris, blanco, az'
```

# 3 Estrategias de generación

In [ ]:
```python
from transformers import AutoModelForCausalLM, AutoTokenizer; model_name = "Qwen/Qwen3-0.6B"
model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype="auto", device_map="auto")
tokenizer = AutoTokenizer.from_pretrained(model_name) # padding_side="left"
inputs = tokenizer(["Lista de colores: rojo, azul"], return_tensors="pt").to(model.device)
```

**Métodos de decodificación básicos:** greedy, sampling y beam search

- **Greedy search:** el siguiente token es el más probable de la distribución
- **Sampling:** el siguiente token se muestrea de la distribución
- **Beam search:** la siguiente secuencia de tokens es la más probable de las exploradas con un haz limitado

In [ ]:
```python
outputs = model.generate(**inputs, max_new_tokens=50)
print("\nGreedy:\n",  tokenizer.batch_decode(outputs, skip_special_tokens=True)[0])
outputs = model.generate(**inputs, max_new_tokens=50, do_sample=True, num_beams=1)
print("\nSampling:\n",tokenizer.batch_decode(outputs, skip_special_tokens=True)[0])
outputs = model.generate(**inputs, max_new_tokens=50, num_beams=2)
print("\nBeam search:\n", tokenizer.batch_decode(outputs, skip_special_tokens=True)[0])
```

```
Greedy:
 Lista de colores: rojo, azul, verde, blanco, negra, marrón, rojo, violeta, gris, lila, amarillo, dorado, blanco, na
ranja, rosa, marrón, naranja, rosa, marr

Sampling:
 Lista de colores: rojo, azul, verde, blanco, naranja, amarillo, marrón, gris, verde

Listas de colores: rojo, azul, verde, blanco, naranja, amarillo, marrón, gris

Beam search:
 Lista de colores: rojo, azul, verde, blanco, naranja, marrón, amarillo, rojo, verde, azul, blanco, naranja, marrón,
amarillo, rojo, verde, azul, blanco, n
```

**Métodos de decodificación avanzados:** speculative decoding, prompt lookup decoding, etc.

**Métodos de decodificación personalizados:** para desarrollar decodificadores de comportamiento especializado

# 4 Ingeniería de prompts

**Prompting:** consiste en diseñar prompts que ayuden a producir los resultados deseados

**Ejemplo:**

In [ ]:
```
from transformers import pipeline
pipeline = pipeline(model="Qwen/Qwen3-4B", device_map="auto")
prompt = """
Tarea: Clasifica el texto en positivo o negativo.
Texto: Es mi película favorita.
Resultado:"""
print(pipeline(prompt, max_new_tokens=3, do_sample=True, top_k=10)[0]['generated_text'])
```

```
Loading checkpoint shards:   0%|          | 0/3 [00:00<?, ?it/s]
Tarea: Clasifica el texto en positivo o negativo.
Texto: Es mi película favorita.
Resultado: Positivo
```

**Zero-shot prompting:** consiste en usar un buen prompt sin más

In [ ]:
```
prompt = """
Texto: El primer debate presidencial televisado en Estados Unidos tuvo lugar el 28 de septiembre de 1960.
Fecha:"""
print(pipeline(prompt, max_new_tokens=12, do_sample=True, top_k=10)[0]['generated_text'])
```

```
Texto: El primer debate presidencial televisado en Estados Unidos tuvo lugar el 28 de septiembre de 1960.
Fecha: 28 de septiembre de 1960
```

**Few-shot prompting:** consiste en usar ejemplos específicos de generación

In [ ]:
```
prompt = """
Texto: El primer humano que salió al espacio y orbitó la Tierra lo hizo el 12 de abril de 1961.
Fecha: 12-04-1961
Texto: El primer debate presidencial televisado en Estados Unidos tuvo lugar el 28 de septiembre de 1960.
Fecha:"""
print(pipeline(prompt, max_new_tokens=12, do_sample=True, top_k=10)[0]['generated_text'])
```

```
Texto: El primer humano que salió al espacio y orbitó la Tierra lo hizo el 12 de abril de 1961.
Fecha: 12-04-1961
Texto: El primer debate presidencial televisado en Estados Unidos tuvo lugar el 28 de septiembre de 1960.
Fecha: 28-09-1960
```

**Chain-of-thought (CoT):** consiste en emplear una serie de prompts de razonamiento que ayuden al modelo a "pensar" para hallar la respuesta deseada

In [ ]:
```
prompt = """Let's go through this step-by-step:
1. You start with 15 muffins.
2. You eat 2 muffins, leaving you with 13 muffins.
3. You give 5 muffins to your neighbor, leaving you with 8 muffins.
4. Your partner buys 6 more muffins, bringing the total number of muffins to 14.
5. Your partner eats 2 muffins, leaving you with 12 muffins.
If you eat 6 muffins, how many are left?"""
print(pipeline(prompt, max_new_tokens=12, num_beams=10)[0]['generated_text'])
```

```
Let's go through this step-by-step:
1. You start with 15 muffins.
2. You eat 2 muffins, leaving you with 13 muffins.
3. You give 5 muffins to your neighbor, leaving you with 8 muffins.
4. Your partner buys 6 more muffins, bringing the total number of muffins to 14.
5. Your partner eats 2 muffins, leaving you with 12 muffins.
If you eat 6 muffins, how many are left? 12 - 6 = 6 muffins left
```

# 5 Benchmarking

**LLM benchmarking:** comparación de LLMs con colecciones de tareas

- **HF leaderboards:** doc HF sobre clasificaciones y evaluaciones
- **Artificial Analysis:** análisis del estado actual de la IA y LLMs en particular; consulta el LLM Leaderboard
- **LiveBench:** LLM leaderboard sin "contaminación" del test set

**Measuring Massive Multitask Language Understanding (MMLU):** benckmark de búsqueda de respuestas a preguntas de múltiples tareas (dominios)

**MMLU-Pro:** versión extendida de MMLU

**MMLU-ProX:** versión multilíngüe de MMLU-Pro

**Ejercicio:** prueba algún modelo Qwen3 con alguna pregunta en castellano del test de MMLU-ProX

In [ ]:
```
from datasets import load_dataset; from transformers import pipeline
pipe = pipeline(model="Qwen/Qwen3-4B-FP8", device_map="auto")
ds = load_dataset("li-lab/MMLU-ProX", "es", split="test")
```

**Solución:**

In [ ]:
```
prompt = """
Texto: Los organismos reguladores de publicidad típicos sugieren, por ejemplo, que los anuncios no deben: fomentar _
A: Prácticas seguras, Miedo, Celos, Trivial
B: Prácticas inseguras, Angustia, Alegría, Trivial
C: Prácticas seguras, Deseos, Celos, Trivial
D: Prácticas seguras, Angustia, Miedo, Trivial
E: Prácticas inseguras, Deseos, Celos, Trivial
F: Prácticas seguras, Angustia, Celos, Trivial
G: Prácticas seguras, Deseos, Miedo, Trivial
H: Prácticas inseguras, Deseos, Miedo, Trivial
I: Prácticas inseguras, Angustia, Miedo, Trivial
Tarea: indica con una sola letra la opción correcta, sin explicación
Respuesta:
"""
print(pipe(prompt, max_new_tokens=2, num_beams=2)[0]['generated_text'])
```

```
Texto: Los organismos reguladores de publicidad típicos sugieren, por ejemplo, que los anuncios no deben: fomentar _
________, causar ________ o _____ innecesarios, y no deben causar ofensa _______.
A: Prácticas seguras, Miedo, Celos, Trivial
B: Prácticas inseguras, Angustia, Alegría, Trivial
C: Prácticas seguras, Deseos, Celos, Trivial
D: Prácticas seguras, Angustia, Miedo, Trivial
E: Prácticas inseguras, Deseos, Celos, Trivial
F: Prácticas seguras, Angustia, Celos, Trivial
G: Prácticas seguras, Deseos, Miedo, Trivial
H: Prácticas inseguras, Deseos, Miedo, Trivial
I: Prácticas inseguras, Angustia, Miedo, Trivial
Tarea: indica con una sola letra la opción correcta, sin explicación
Respuesta:
F
```