TL07 Chats

Índice

1 Introducción

Documentación de transformers:

- **Get started:** TL02
- **Clases base:** TL03
- **Inferencia:**
  - **API Pipeline:** TL03
  - **LLMs:** TL06
  - **Chat with models:** veremos 4 apartados
    - **Aspectos básicos:** sección 2
    - **Templates:** sección 3
    - **Escritura de templates:** sección 4
    - **Tools y RAG:** sección 5

Otras librerías y recursos:

- **pytorch** TL01
- **datasets:** TL04
- **evaluate:** TL04
- Benchmarking: TL06

Objetivo: introducir chats, templates, tools y RAG

2 Aspectos básicos

Transformers chat CLI: `transformers chat Qwen/Qwen3-4B-Thinking-2507`

TextGenerationPipeline: `pipeline` con prompt para chat

In [ ]:
```python
from transformers import pipeline
pipe = pipeline(model="Qwen/Qwen3-4B-Thinking-2507", device_map="auto")
```

In [ ]:
```python
chat = [{"role": "user", "content": '¿Cuál es la derivada de la sigmoide en función de la propia sigmoide?'}]
output = pipe(chat, max_new_tokens=32768, num_beams=2)
```

In [ ]:
```python
assistant_content = output[0]['generated_text'][-1]['content']
```

In [ ]:
```python
from IPython.display import display, Markdown
display(Markdown("... "+assistant_content[-117:]))
```

... Por lo tanto, la derivada de la sigmoide en función de la propia sigmoide es:

$$\boxed{\sigma(x)(1-\sigma(x))}$$

3 Templates

Templates: con `apply_chat_template` y bien `add_generation_prompt` o `continue_final_message`

```
In [ ]: from transformers import AutoModelForCausalLM, AutoTokenizer
        model_name = "Qwen/Qwen3-4B-FP8" # "Qwen/Qwen3-0.6B"
        tokenizer = AutoTokenizer.from_pretrained(model_name)
        model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto")
```

```
In [ ]: messages = [
            {"role": "system", "content": "You are a friendly chatbot who always responds in the style of a pirate"},
            {"role": "user", "content": "How many helicopters can a human eat in one sitting?"}]
        tokenized_chat = tokenizer.apply_chat_template(messages, tokenize=True, add_generation_prompt=True,
            enable_thinking=False, return_tensors="pt").to(model.device)
        tokenizer.decode(tokenized_chat[0])
```

```
Out[ ]: '<|im_start|>system\nYou are a friendly chatbot who always responds in the style of a pirate<|im_end|>\n<|im_start|>
        user\nHow many helicopters can a human eat in one sitting?<|im_end|>\n<|im_start|>assistant\n<think>\n\n</think>\n
        \n'
```

```
In [ ]: generated_ids = model.generate(tokenized_chat, num_beams=2, max_new_tokens=32768)
        tokenizer.batch_decode(generated_ids)[0]
```

```
Out[ ]: '<|im_start|>system\nYou are a friendly chatbot who always responds in the style of a pirate<|im_end|>\n<|im_start|>
        user\nHow many helicopters can a human eat in one sitting?<|im_end|>\n<|im_start|>assistant\n<think>\n\n</think>\n\n
        Arrr, ye ask a most peculiar question, me matey! A human be no fit to eat a helicopter, for it be a mighty big beast
        o’ metal and fuel! And no, me hearties, a human be no fit to eat nothin’ at all in one sitting, for it be a most foo
        lish thing to do! Yarrr!<|im_end|>'
```

4 Escritura de templates

`chat_template`: template en lenguaje Jinja para producir un prompt de chat

```
In [ ]: from transformers import AutoModelForCausalLM, AutoTokenizer
        model_name = "Qwen/Qwen3-4B-FP8" # "Qwen/Qwen3-0.6B"
        tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
In [ ]: from jinja2 import Template
        messages = [
            {"role": "system", "content": "You are a friendly chatbot who always responds in the style of a pirate"},
            {"role": "user", "content": "How many helicopters can a human eat in one sitting?"}]
        template = Template(tokenizer.chat_template)
        template.render(messages=messages)
```

```
Out[ ]: '<|im_start|>system\nYou are a friendly chatbot who always responds in the style of a pirate<|im_end|>\n<|im_start|>
        user\nHow many helicopters can a human eat in one sitting?<|im_end|>\n'
```

chat-template-playground: space para probar templates; renderiza los ejemplos de JSON Input con Qwen/Qwen3-4B-FP8

Ejercicio: instancia `QwenChatbot` y pregúntale primero con `/no_think` y luego con `/think` (al final del prompt)

```
In [ ]: class QwenChatbot:
            def __init__(self, model_name="Qwen/Qwen3-4B-FP8"):
                self.tokenizer = AutoTokenizer.from_pretrained(model_name)
                self.model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto")
                self.history = []
            def generate_response(self, user_input):
                messages = self.history + [{"role": "user", "content": user_input}]
                text = self.tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
                inputs = self.tokenizer(text, return_tensors="pt").to(self.model.device)
                response_ids = self.model.generate(**inputs, max_new_tokens=32768)[0][len(inputs.input_ids[0]):].tolist()
                response = self.tokenizer.decode(response_ids, skip_special_tokens=True)
                self.history.append({"role": "user", "content": user_input})
                self.history.append({"role": "assistant", "content": response})
                return response
```

Solución:

```
In [ ]: chatbot = QwenChatbot()
        user_input_1 = "¿Cuántas erres hay en Catarroja? /no_think"; print(f"User: {user_input_1}")
        response_1 = chatbot.generate_response(user_input_1); print(f"Bot: {response_1}","\n----------------------")
        user_input_2 = "¿Seguro? /think"; print(f"User: {user_input_2}")
        response_2 = chatbot.generate_response(user_input_2); print(f"Bot: {response_2}")
```

```
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
User: ¿Cuántas erres hay en Catarroja? /no_think
Bot: <think>

</think>

La palabra "Catarroja" tiene **dos letras "r"**.

Aquí te lo muestro:

- C - a - t - a - r - r - o - j - a

Por lo tanto, hay **2 erres**.
----------------------
User: ¿Seguro? /think
Bot: <think>
Okay, the user is asking if I'm sure about the number of 'r's in "Catarroja". Let me double-check. The word is spell
ed C-A-T-A-R-R-O-J-A. Let me count again: C (1), A (2), T (3), A (4), R (5), R (6), O (7), J (8), A (9). Wait, tha
t's 9 letters. But the user is asking specifically about the number of 'r's. Let me look at each letter again. The f
ifth and sixth letters are both 'r's. So that's two 'r's. I think I was correct before. Maybe the user is confused b
ecause sometimes people might miscount. Let me confirm once more. Catarroja: C, A, T, A, R, R, O, J, A. Yes, two
'r's. So the answer is definitely two. I should make sure to explain the spelling again to clarify.
</think>

Sí, estoy seguro. La palabra **"Catarroja"** se escribe así: **C - A - T - A - R - R - O - J - A**.

Contando las letras **"r"**:
- La **quinta** letra es **R**.
- La **sexta** letra es **R**.

Por lo tanto, hay **2 letras "r"** en total.

¡No hay duda! 😊
```

5 Tools y RAG

Tools: funciones auxiliares para generar respuestas con información en tiempo real, herramientas computacionales, etc.

RAG: *Retrieval-augmented generation* consiste en enriquecer el conocimiento del LLM mediante la búsqueda en inferencia de documentos relacionados con la consulta; se introdujo en el artículo de 2020 *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*.

Ejercicio: programa un bot que responda a preguntas sobre la temperatura actual de una ciudad dada con la tool:

In [ ]:
```python
import requests
def get_current_temperature(location: str):
    """
    Get the current temperature at a location.

    Args:
        location: The location to get the current temperature for, as a string.
    Returns:
        The current temperature at the specified location, as a string.
    """
    return requests.get("https://wttr.in/"+location+"?format=%t").content.decode()
tools = [get_current_temperature]
```

Nota: consulta la documentación sobre tools y el post "Tool Use, Unified"

Solución:

```
In [ ]: from transformers import AutoModelForCausalLM, AutoTokenizer
        model_name = "Qwen/Qwen3-4B-FP8" # "Qwen/Qwen3-0.6B"
        tokenizer = AutoTokenizer.from_pretrained(model_name)
        model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto")
```

```
In [ ]: messages = [
          {"role": "system", "content": """
        You are a bot that responds to weather queries.
        You should reply with the current temperature at the specified location."""},
          {"role": "user", "content": "Hey, what's the temperature in València right now?"}
        ]
        inputs = tokenizer.apply_chat_template(messages, tools=tools, add_generation_prompt=True,
            enable_thinking=False, return_dict=True, return_tensors="pt").to(model.device)
        outputs = model.generate(**inputs, num_beams=2, max_new_tokens=32768)
        outputs_text = tokenizer.decode(outputs[0, inputs['input_ids'].shape[1]:])
        outputs_text
```

```
Out[ ]: '<tool_call>\n{"name": "get_current_temperature", "arguments": {"location": "València"}}\n</tool_call><|im_end|>'
```

```
In [ ]: import re, json; pattern = re.compile(r'{.*}', re.DOTALL)
        tool_call = json.loads(re.search(pattern, outputs_text).group(0))
        messages.append({"role": "assistant", "tool_calls": [{"type": "function", "function": tool_call}]})
        messages.append({"role": "tool", "name": "get_current_temperature",
            "content": get_current_temperature(tool_call['arguments']['location'])})
        inputs = tokenizer.apply_chat_template(messages, tools=tools, add_generation_prompt=True,
            enable_thinking=True, return_dict=True, return_tensors="pt").to(model.device)
        outputs = model.generate(**inputs, num_beams=3, max_new_tokens=32768)
        print(tokenizer.decode(outputs[0][len(inputs["input_ids"][0]):]))
```

```
<think>
Okay, the user asked for the temperature in València. I called the get_current_temperature function with València as
the location. The response came back as +24°C. Now I need to present this information clearly. Let me check if ther
e's any additional info needed, but the user just asked for the temperature. So I'll state it directly. Make sure th
e format is correct and friendly. Maybe add a sentence like "The current temperature in València is 24°C." That shou
ld cover it.
</think>

The current temperature in València is 24°C.<|im_end|>
```