TL03 pipeline

Índice

1 Introducción

Documentación de transformers:

- **Get started:** TL02
- **Clases base:**
  - **Modelos:** sección 2
  - **Preprocesadores:** sección 3
- **Inferencia:**
  - **API Pipeline:** sección 4
  - ...

Otras librerías y recursos:

- **pytorch** TL01

Objetivo: introducir clases base, API Pipeline y aplicaciones

2 Modelos

Exploración de modelos: en HF y github

Ejercicio: halla y describe brevemente el propósito de los modelos más descargados

Documentación de transformers: clases base $\longrightarrow$ modelos

- **Loading models:** `AutoModel` (barebone o con cabeza) y `from_pretrained()`
- **Customizing models:** a partir de `modeling.py` y `configuration.py`
- **Customizing model components:** cargamos, modificamos, `clear_import_cache()` y recargamos
- **Sharing:** de modelos en el hub HF con cuenta y token de acceso
- **Adding a new model to Transformers:** buenas prácticas, arquitectura de los modelos, principios de diseño, etc.
- **Modular Transformers:** filosofía de diseño de transformers
- **Document your models:** decorador `@auto_docstring`
- **Customizing attention function:** `AttentionInterface`

Solución (julio de 2025):

1. Falconsai/nsfw_image_detection: clasificación de imágenes NSFW (Not Safe for Work) para filtrar contenido inapropiado (en entrenamiento e inferencia)
2. sentence-transformers/all-MiniLM-L6-v2: transformación de frases y párrafos a un espacio 384-dimensional para tareas como clustering o búsqueda semántica
3. timm/mobilenetv3_small_100.lamb_in1k: modelo ligero de clasificación de imágenes para móviles
4. dima806/fairface_age_image_detection: clasificación de imágenes de caras en grupos de edad
5. google-bert/bert-base-uncased: BERT en inglés con minúsculas para masked language modeling (MLM), next sentence prediction (NSP) y, sobre todo, para adaptación a tareas específicas como clasificación de secuencias, clasificación de tokens o búsqueda de respuestas
6. amazon/chronos-t5-small: familia de modelos pequeños de predicción de series temporales basada en la arquitectura T5
7. facebook/esm2_t30_150M_UR50D: MLM de proteinas adaptable a tareas específicas
8. Bingsu/adetailer: detección de caras, manos, personas y prendas de ropa
9. timm/resnet50.a1_in1k: red convolucional refinada para clasificación de imágenes
10. sentence-transformers/all-mpnet-base-v2: transformación de frases y párrafos a un espacio 768-dimensional
11. pyannote/wespeaker-voxceleb-resnet34-LM: embedding de locutores de pyannote
12. pyannote/segmentation-3.0: diarización de (hasta tres) locutores a partir de 10 segundos de audio mono a 16kHz
13. openai/clip-vit-base-patch32: clasificación de imágenes zero-shot (CLIP=Contrastive Language–Image Pre-training)
14. openai-community/gpt2: versión más pequeña de GPT-2 para generación de texto en inglés o adaptación a tareas específicas (no recomendado para interacción con personas ya que se entrenó con contenido no filtrado)
15. google/electra-base-discriminator: alternativa eficiente al pretraining MLM convencional (BERT)

3 Preprocesadores

Documentación de tokenizers: librería usada por transformers

- **Getting started:**
  - Tokenizers: características principales
  - Quicktour: entrenamiento, uso, postproceso, batching, modelos preentrenados
  - The tokenization pipeline: normalización, pretokenización, modelo, postproceso
  - Components: normalizadores, pretokenizadores, modelos, postprocesadores, decoders
  - Training from memory: con ficheros de texto, usando la librería datasets o con ficheros gzip
- **API:** ...

Documentación de transformers: clases base $\longrightarrow$ preprocesadores

- **Tokenizers:** `AutoTokenizer` y `from_pretrained()`
- **Image processors:** `AutoImageProcessor` y `from_pretrained()`
- **Video processors:** `AutoVideoProcessor` y `from_pretrained()`
- **Backbones:** para visión con `AutoBackbone` y `from_pretrained()`
- **Feature extractors:** para audio con `AutoFeatureExtractor` y `from_pretrained()`
- **Processors:** para multimodal con `AutoProcessor` y `from_pretrained()`
- **Summary of the tokenizers:** BPE, BBPE, WordPiece, Unigram, SentencePice
- **Padding and truncation :** parámetros `padding`, `truncation` y `max_length` del tokenizador

4 Pipeline

Documentación de transformers: inferencia $\longrightarrow$ API Pipeline

- **Pipeline:** método sencillo para inferencia; basta indicar una tarea y, opcionalmente, un modelo preentrenado
- **Machine learning apps:** con gradio y `pipeline()`
- **Web server inference:** con starlette, uvicorn y `pipeline()`
- **Adding a new pipeline:** mediante instanciación de `Pipeline` e implementación de unos pocos métodos

Tareas y modelos:

- **Exploración en HF:** tasks o models (filtrando por tareas)
- **Tags de un modelo dado:** p. e. `text-classification` de tabularisai/multilingual-sentiment-analysis

Ejemplo: análisis de opinión multilíngüe con tabularisai/multilingual-sentiment-analysis

In [ ]:
```
from transformers import pipeline; model_name = "tabularisai/multilingual-sentiment-analysis"
pipe = pipeline(task="text-classification", model=model_name, device_map="auto")
pipe(["Me encanta este producto!", "Me he dormido viendo la película."])
```

Out[ ]:
```
[{'label': 'Very Positive', 'score': 0.5204034447669983},
 {'label': 'Negative', 'score': 0.41417378187179565}]
```

5 Ejercicio

Prueba un ejemplo de inferencia con `pipeline` para cada una de las siguientes tareas y modelos:

- Búsqueda de respuestas multilíngüe: `question-answering` con `timpal0l/mdeberta-v3-base-squad2`
- Named entity recognition (NER) en castellano: `token-classification` con `MMG/xlm-roberta-large-ner-spanish`
- Traducción automática multilíngüe: `translation` con `facebook/nllb-200-distilled-600M`
- Resumen en castellano o valenciano: `summarization` con `ELiRF/NASES`
- Masked language modeling: `fill-mask` con `bert-base-multilingual-cased`
- Next sentence prediction: `NextSentencePrediction` con `bert-base-multilingual-cased` (no con `pipeline`; ver API de BERT)

6 Solución

Búsqueda de respuestas multilíngüe:

In [ ]:
```
from transformers import pipeline; model_name = "timpal0l/mdeberta-v3-base-squad2"
pipe = pipeline(task="question-answering", model=model_name, device_map="auto")
pipe(question="Dónde vive?", context="Se llama Jorge y reside en València.")
```

Out[ ]:
```
{'score': 0.9710777997970581, 'start': 26, 'end': 36, 'answer': ' València.'}
```

Named entity recognition (NER) en castellano:

In [ ]:
```
from transformers import pipeline; model_name = "MMG/xlm-roberta-large-ner-spanish"
pipe = pipeline(task="token-classification", model=model_name, device_map="auto")
pipe("José Capilla es el rector de la UPV.", aggregation_strategy="simple")
```

Out[ ]:
```
[{'entity_group': 'PER',
  'score': 0.9947879,
  'word': 'José Capilla',
  'start': 0,
  'end': 12},
 {'entity_group': 'ORG',
  'score': 0.99299836,
  'word': 'UPV',
  'start': 32,
  'end': 35}]
```

Traducción automática multilíngüe:

In [ ]:
```
from transformers import pipeline; model_name = "facebook/nllb-200-distilled-600M"
pipe = pipeline(task="translation", model=model_name)
frase_eng = "How extraordinary rainfall caught Texas by surprise"
print(pipe(frase_eng, src_lang="eng", tgt_lang="spa_Latn")[0]['translation_text'])
print(pipe(frase_eng, src_lang="eng", tgt_lang="cat_Latn")[0]['translation_text'])
```

```
Cómo una lluvia extraordinaria atrapó a Texas por sorpresa.
Com les precipitacions extraordinàries van sorprendre a Texas?
```

Resumen en castellano o valenciano:

In [ ]:
```
from transformers import pipeline; model_name = "ELiRF/NASES"
pipe = pipeline(task="summarization", model=model_name, device_map="auto")
pipe('''El mes de junio más tórrido desde que hay registros ha dejado en España un calor extremo y
temperaturas récord, con máximas y mínimas varios grados por encima de la media histórica. Por si esto fuera poco,
después de un mes lo suficientemente asfixiante, el 28 de junio se desencadenó una ola de calor especialmente
intensa, con temperaturas más propias de julio y agosto y un récord negativo, los 46 grados que se alcanzaron
en el pueblo de El Granado (Huelva), algo que no había ocurrido nunca.
La anomalía climática no solo asusta, sino que también tiene consecuencias claras y directas: el pasado mes de
junio murieron 407 personas por el fuerte calor en España, según el sistema de monitorización de la mortalidad
diaria por todas las causas (MoMo), elaborado por el Instituto de Salud Carlos III. Una cifra que sitúa al país
a la cabeza de Europa en esta triste estadística. El dato es muy llamativo, sobre todo porque es muy superior al
registrado en junio de 2024, cuando murieron 32 personas por las altas temperaturas.
El calor fue especialmente letal en los tres últimos días del mes: solo entre el sábado 28 y el domingo 30 de
junio  se contabilizaron 102 muertes relacionadas directamente con las altas temperaturas. El 30 de junio fue
el día más mortífero, con 46 muertes.''')[0]['summary_text']
```

Out[ ]:
```
'El pasado mes de junio murieron 407 personas por el fuerte calor en España, según el sistema de monitorización de l
a mortalidad.'
```

In [ ]:
```
pipe('''La calor marca un nou cap de setmana a la Comunitat Valenciana, encara que este dissabte compartirà
protagonisme amb les tempestes, que poden descarregar aigua amb força a l'interior de València i Castelló
i anar acompanyades de granís, pedra i ràfegues de vent intenses. Amb este oratge, l'Agència Estatal de
Meteorologia ha decretat avís de nivell groc en bona part del territori, que afectarà les tres províncies,
encara que per casuístiques diferents.
D'una banda, estan les temperatures altes, que han encés l'avís en tot el litoral de València, l'interior sud
d'esta província i també la costa des de la Marina Baixa fins al Baix Segura. La matinada ja ha sigut sufocant,
amb termòmetres que no han baixat dels 25 °C, i la previsió apunta que este dissabte encara farà més basca que
ahir. Passades les huit del matí ja se superaven els 30 °C en molts punts a vora mar, i s'esperen màximes entorn
dels 36 °C, que poden acostar-se als 40 °C en els pobles de l'interior sud de la província de València. Encara
que el dia ha trencat assolellat, de cara a la vesprada creixeran els cúmuls de núvols, que poden descarregar
precipitacions i alguna tempesta. Al llarg del matí es pot escapar algun ruixat cap al nord,  camí a Castelló,
però serà al migdia quan les tempestes cobraran força. Poden anar acompanyades de granissades i  pedregades amb
ratxes de  vent molt fortes, i per això l'Aemet ha activat l'avís groc a l'interior de les províncies de València
i Castelló. Tot i això, al final de la jornada pot arribar algun ruixat al litoral.''')[0]['summary_text']
```

Out[ ]:
```
'La temperatura marca un nou cap de setmana a la Comunitat Valenciana, encara que este dissabte compartirà protagoni
sme amb les tempestes.'
```

Masked language modeling:

In [ ]:
```
from transformers import pipeline; model_name = 'bert-base-multilingual-cased'
pipe = pipeline('fill-mask', model=model_name); pipe("Hello I'm a [MASK] model.")
```

Out[ ]:
```
[{'score': 0.10182151943445206,
  'token': 13192,
  'token_str': 'model',
  'sequence': "Hello I ' m a model model."},
 {'score': 0.05212624743580818,
  'token': 11356,
  'token_str': 'world',
  'sequence': "Hello I ' m a world model."},
 {'score': 0.048930175602436066,
  'token': 11165,
  'token_str': 'data',
  'sequence': "Hello I ' m a data model."},
 {'score': 0.020360128954052925,
  'token': 23578,
  'token_str': 'flight',
  'sequence': "Hello I ' m a flight model."},
 {'score': 0.0200796015560627,
  'token': 14155,
  'token_str': 'business',
  'sequence': "Hello I ' m a business model."}]
```

Next sentence prediction: 0 si sí es continuación; 1 si no

In [ ]:
```
from transformers import AutoTokenizer, BertForNextSentencePrediction; import torch
model = BertForNextSentencePrediction.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
fraseA = "In Italy, pizza served in formal settings, such as at a restaurant, is presented unsliced."
fraseB = "The sky is blue due to the shorter wavelength of blue light."
encoding = tokenizer(fraseA, fraseB, return_tensors="pt")
print(model(**encoding, labels=torch.LongTensor([1])).logits.argmax().item())
fraseB = "So you can cut slices at your preference."
encoding = tokenizer(fraseA, fraseB, return_tensors="pt")
print(model(**encoding, labels=torch.LongTensor([1])).logits.argmax().item())
```

```
1
0
```