Clase 145 — Hugging Face Transformers (uso práctico)

Parte: 2 — Deep Learning · Fuente: HuggingFace docs + Géron, cap. 16 § Pretrained Transformer Models. Duración estimada: 80 min.

Clase 145 — Hugging Face Transformers (uso práctico)

> *Parte: 2 — Deep Learning · Fuente: HuggingFace docs + Géron, cap. 16 § Pretrained Transformer Models. Duración estimada: 80 min.*

Objetivo

Dominar Hugging Face Transformers — la librería estándar de la industria para usar modelos preentrenados (BERT, GPT, T5, Llama, Whisper, ViT...). Aprender el pipeline API (one-liner para 90 % de los casos), el Trainer API (fine-tuning con muy poco código) y los componentes manuales (Tokenizer + Model + DataLoader).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Usar pipeline('sentiment-analysis'), pipeline('summarization'), pipeline('zero-shot-classification'), etc. en 1 línea.
- Cargar manualmente: tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased'), model = AutoModelForSequenceClassification.from_pretrained('...').
- Tokenizar con tokenizer(texts, padding=True, truncation=True, return_tensors='pt').
- Fine-tunear con Trainer sobre un dataset propio.
- Reconocer el Hub (huggingface.co/models) como catálogo de + de 500k modelos.

Temas

- pipeline API: lo más fácil. Tareas: sentiment, NER, QA, summarization, translation, fill-mask, zero-shot, etc.
- AutoTokenizer + AutoModel*: API manual flexible.
- Tokenizers: BPE, WordPiece, SentencePiece, tiktoken.
- Trainer + TrainingArguments: fine-tuning con 20 líneas.
- Hub: descubrir modelos, datasets, spaces.
- datasets library: cargar datasets, preprocesar.

Definiciones y características

- AutoTokenizer: detecta el tokenizer adecuado para el modelo.
- AutoModelFor<Task>: cabezas específicas (SequenceClassification, TokenClassification, CausalLM, Seq2SeqLM).
- Subword tokenization (BPE/WP/SP): tokens son piezas de palabras → vocab fijo (~32-64k), nada de OOV.
- Special tokens: [CLS], [SEP], <s>, </s>, [PAD], <|im_start|>, etc. Dependen del modelo.
- Hub: GitHub para modelos — cualquiera puede subir/descargar.

Dataset / recursos

- HuggingFace Hub: <https://huggingface.co/models>.
- Dataset: load_dataset('imdb'), load_dataset('squad'), etc.
- Librerías: transformers, datasets, accelerate, evaluate.

Ejercicios

1. pipeline one-liner: pipe = pipeline('sentiment-analysis'); pipe('I loved this movie!'). Inspeccionar output.
2. Zero-shot classification: pipeline('zero-shot-classification')(text, candidate_labels=['sports', 'politics', 'tech']). Sin training específico.
3. Manual: tokenizar texto con bert-base-uncased. Inspeccionar input_ids, attention_mask, token_type_ids.
4. Modelo + forward: outputs = model(**inputs). Inspeccionar outputs.logits.
5. Tokenizer especiales: tokenizer.decode([101, 7592, 102]) → '[CLS] hello [SEP]'.

Homework verificable

Fine-tuning de distilbert-base-uncased sobre IMDB:

1. load_dataset('imdb').
2. Tokenizar con tokenizer(texts, truncation=True, padding='max_length', max_length=256).
3. AutoModelForSequenceClassification.from_pretrained(..., num_labels=2).
4. TrainingArguments(output_dir, num_train_epochs=2, per_device_train_batch_size=16, eval_strategy='epoch').
5. Trainer(...).train().
6. Reportar accuracy en test.

Criterio de aceptación: accuracy ≥ 0.92 (DistilBERT con 2 épocas alcanza ~0.93 en IMDB).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
OSError: Can't load tokenizer	Modelo no existe o sin internet. Fix: veri
RuntimeError: CUDA out of memory	Modelo + batch demasiado grandes. Fix: red
Tokenizer no agrega special tokens automát	Pasar add_special_tokens=True (default). S
Trainer no usa GPU	Default lo detecta; sino TrainingArguments
Fine-tuning de un modelo grande catastroph	LR alto. Fix: learning_rate=2e-5 o menor p

Preguntas frecuentes

¿pipeline o Trainer o AutoModel?

pipeline para usar un modelo preentrenado tal cual. AutoModel para custom training loop. Trainer para fine-tuning estándar (95 % de los casos).

¿PyTorch o TF backend?

PyTorch es estándar en HF — más modelos disponibles, mejor soportado. transformers soporta ambos pero los modelos modernos (Llama, Mistral) son solo PyTorch.

¿Tokenizer rápido?

AutoTokenizer.from_pretrained(..., use_fast=True) (default). Implementado en Rust, 10-100× más rápido que la versión Python.

¿Cómo descubro qué modelo usar?

<https://huggingface.co/models> con filtros por task, language, license. Modelos más descargados suelen ser buena apuesta.

¿Hugging Face Inference API en producción?

Sí, para prototipos. Para producción serio, self-host con transformers o vLLM / TGI (clase 139).

Referencias

- Hugging Face Transformers docs.
- Hugging Face course — el mejor recurso gratuito para empezar.
- Wolf et al. (2020), Transformers: State-of-the-Art Natural Language Processing, EMNLP demo.

Siguiente clase

Clase 146 — CLIP, SigLIP: multimodal embeddings (visión + texto)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb