Ampliación de Fundamentos de Hardware

Angel Fabrizio Ullaguari Yanza

2ºASIR

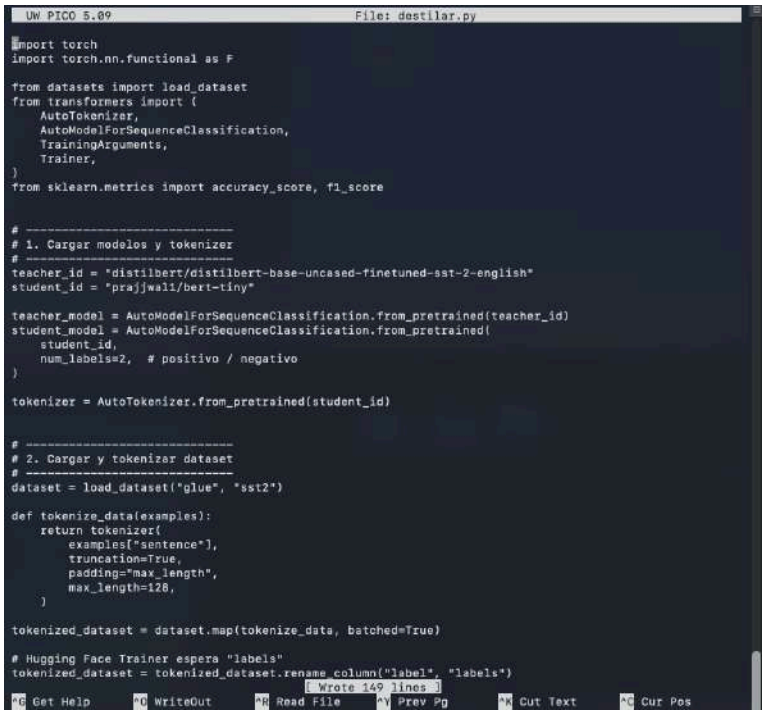
Práctica IA. Destilación

Objetivo

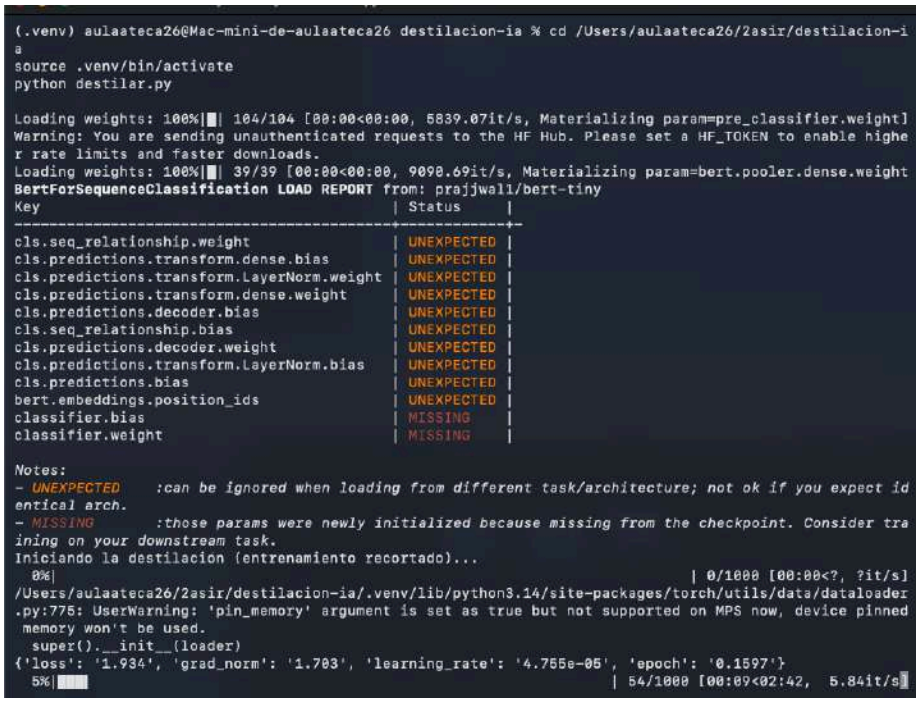
Tomar un modelo profesor (basado en DistilBERT), ya afinado para análisis de sentimientos, y destilar su conocimiento a un modelo estudiante (BERT-tiny) que es ~15 veces más pequeño. El resultado será un clasificador de sentimientos ultraligero y veloz.

Configuración

Para empezar, creamos una carpeta limpia para el proyecto y configuramos un entorno virtual con python, creando así el archivo destilar.py. Con el comando nano ponemos el código puesto en el enunciado, que pide expresamente un modelo de IA y otro más pequeño (profesor y estudiante). Pero antes de crearlo, se **necesita tener instaladas las librerías de Hugging Face (torch, transformers, peft, etc...).**



Después de guardar los cambios, pasamos al destilar nuestro modelo: ejecutamos con python el archivo. Al principio, nos aparecen unos avisos de la librería de Hugging Face pero no hemos tenido ningún problema.



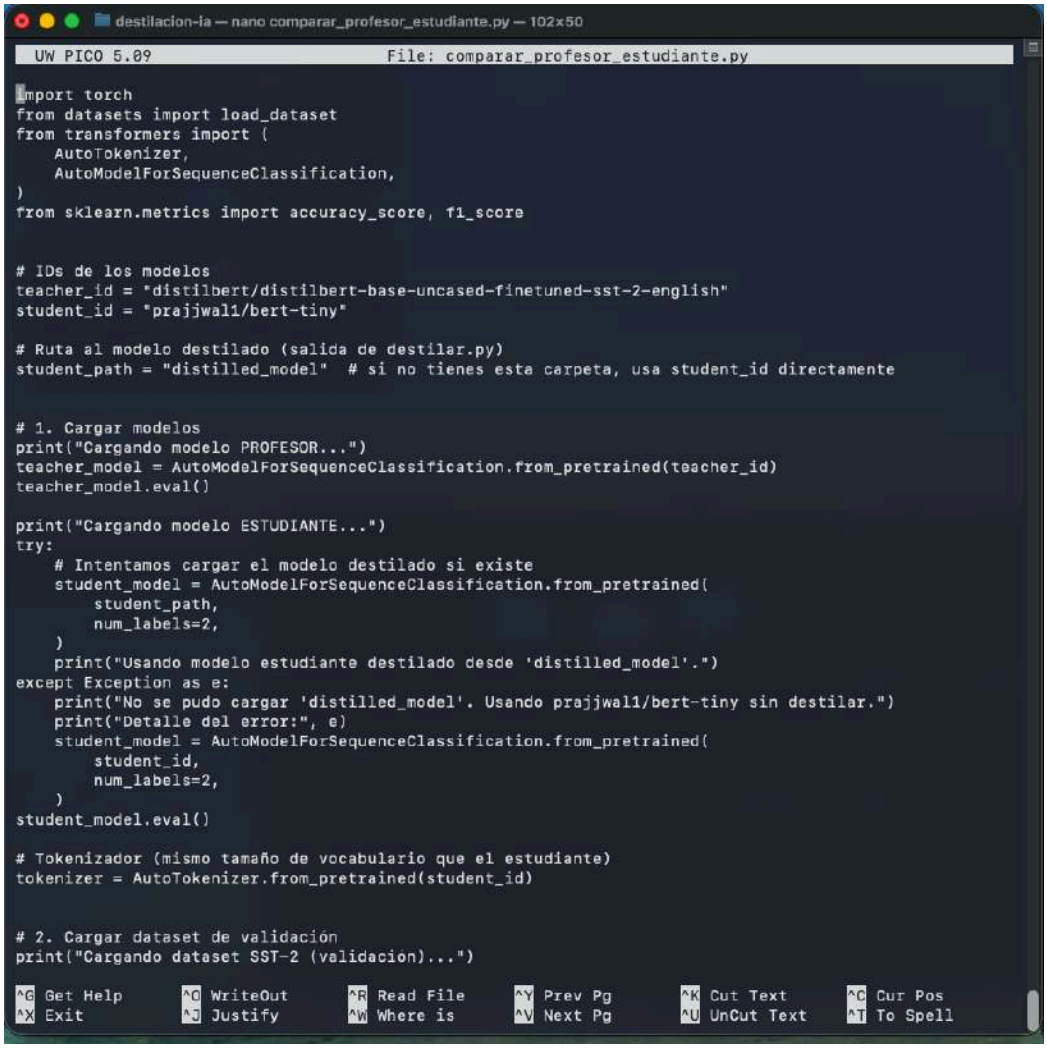
Ahora creamos el archivo probar_modelo.py para ver si el modelo estudiante funciona.

```
  UW PICO 5.09                                File: probar_modelo.py

import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# Usamos directamente el identificador del modelo estudiante
base_id = "prajjwal1/bert-tiny"

# Cargamos el tokenizer y el modelo de Hugging Face
tokenizer = AutoTokenizer.from_pretrained(base_id)
student_model = AutoModelForSequenceClassification.from_pretrained(
    base_id,
    num_labels=2,  # positivo / negativo
)

student_model.eval()


def clasificar(texto: str):
    inputs = tokenizer(
        texto,
        return_tensors="pt",
        truncation=True,
        padding=True,
        max_length=128,
    )
    with torch.no_grad():
        outputs = student_model(**inputs)
        probs = torch.softmax(outputs.logits, dim=-1)[0]
    etiqueta = "positivo" if probs[1] > probs[0] else "negativo"
    print(f"Texto: {texto}")
    print(f"Prediccion: {etiqueta} (probs = {probs.tolist()})\n")


if __name__ == "__main__":
    clasificar("This movie was fantastic, I really loved it!")
    clasificar("This movie was terrible, I will never watch it again.")
```

Lo ejecutamos:

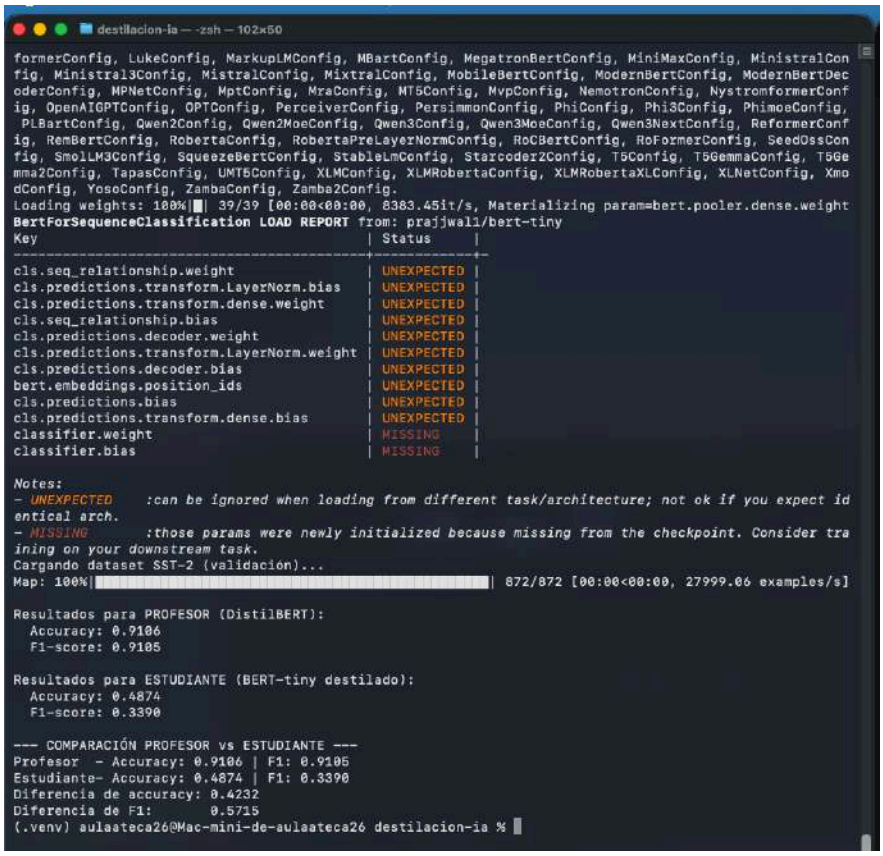
```
(.venv) aulaateca26@Mac-mini-de-aulaateca26 destilacion-ia % cd /Users/aulaateca26/2asir/destilacion-i
a
source .venv/bin/activate
python probar_modelo.py

Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable highe
r rate limits and faster downloads.
Loading weights: 100%|| 39/39 [00:00<00:00, 11080.26it/s, Materializing param=bert.pooler.dense.weigh
BertForSequenceClassification LOAD REPORT from: prajjwal1/bert-tiny
Key                                        | Status     |
-------------------------------------------+------------+-
cls.predictions.transform.dense.bias       | UNEXPECTED |
cls.predictions.decoder.bias               | UNEXPECTED |
cls.seq_relationship.weight                | UNEXPECTED |
cls.predictions.bias                       | UNEXPECTED |
cls.predictions.transform.LayerNorm.weight | UNEXPECTED |
cls.predictions.transform.LayerNorm.bias   | UNEXPECTED |
cls.predictions.decoder.weight             | UNEXPECTED |
bert.embeddings.position_ids               | UNEXPECTED |
cls.predictions.transform.dense.weight     | UNEXPECTED |
cls.seq_relationship.bias                  | UNEXPECTED |
classifier.weight                          | MISSING    |
classifier.bias                            | MISSING    |

Notes:
- UNEXPECTED    :can be ignored when loading from different task/architecture; not ok if you expect id
entical arch.
- MISSING       :those params were newly initialized because missing from the checkpoint. Consider tra
ining on your downstream task.
Texto: This movie was fantastic, I really loved it!
Prediccion: positivo (probs = [0.3990715742111206, 0.6009284257888794])

Texto: This movie was terrible, I will never watch it again.
Prediccion: positivo (probs = [0.3918722867965698, 0.6081277132034302])

(.venv) aulaateca26@Mac-mini-de-aulaateca26 destilacion-ia %
```

Con el entrenamiento finalizado, ahora se debe crear un nuevo archivo .py para comparar las estadísticas del modelo profesor y estudiante.



Comparativa

Ejecutamos el archivo .py:



Al final nos da unos valores que muestran la comparación de precisión y velocidad de cada modelo. El objetivo se ha cumplido, pues se ve que el modelo estudiante da menos parámetros que el modelo profesor.