TL04 evaluate

Índice

1 Introducción

Documentación de transformers:

- **Get started:** TL02
- **Clases base:** TL03
- **Inferencia:**
  - **API Pipeline:** TL03

Otras librerías y recursos:

- **pytorch** TL01
- **datasets:** secciones siguientes
- **evaluate:** secciones siguientes

Objetivo: introducir datasets, evaluate y aplicaciones

2 Datasets

Exploración de datasets: en HF y github

Ejercicio: halla y describe brevemente los (tres primeros) datasets más descargados de más de 1K datos

Documentación de datasets:

- **Get started:** datasets, quickstart e instalación
- **Tutoriales:** overview, carga, exploración, preproceso, creación y compartición
- **Howtos:** uso general, audio, visión, texto, tabular y repos
- **Guías conceptuales:** Arrow, catche, Dataset or IterableDataset, etc.
- **Referencia:** clases principales, builders, loaders, tables y utilidades

Ejemplo: con el dataset nyu-mll/glue; ver quickstart → NLP para más detalles

In [ ]:
```python
from datasets import load_dataset; import torch
from transformers import AutoModelForSequenceClassification, AutoTokenizer
dataset = load_dataset("nyu-mll/glue", "mrpc", split="train")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
def encode(examples):
    return tokenizer(examples["sentence1"], examples["sentence2"], truncation=True, padding="max_length")
dataset = dataset.map(encode, batched=True)
dataset = dataset.map(lambda examples: {"labels": examples["label"]}, batched=True)
dataset = dataset.select_columns(["input_ids", "token_type_ids", "attention_mask", "labels"])
dataset = dataset.with_format(type="torch")
dataloader = torch.utils.data.DataLoader(dataset, batch_size=32)
dataloader
```

Out[ ]:
```
<torch.utils.data.dataloader.DataLoader at 0x766f7142aba0>
```

Solución al ejercicio de exploración (extendido; julio de 2025):

1. permutans/fineweb-bbc-news: 300M de artículos de BBC News (arXiv:2406.1755)
2. SWE-Gym/SWE-Gym: 2438 instancias de 11 repos Python para evaluación de agentes de software (arXiv:2412.21139)
3. nebius/SWE-rebench: dataset para evaluación de agentes de software (arXiv:2505.20411); ver Nebius
4. TAUR-Lab/Taur_CoT_Analysis_Project___gpt-4o-2024-08-06: dataset de TAUR-Lab at UT Austin para analizar GPT-4o
5. HuggingFaceFW/fineweb: dataset de colección de FineData, parte de HF Science
6. Salesforce/wikitext: 100M de tokens de artículos seleccionados de Wikipedia
7. HuggingFaceFW/fineweb-2: análogo a HuggingFaceFW/fineweb
8. adams-story/datacomp200m: versión pequeña del dataset mlfoundations/datacomp_1b de DataComp
9. m-a-p/FineFineWeb: análogo a HuggingFaceFW/fineweb
10. cadene/droid: dataset de robótica con 27M frames, 92K episodios y 31K tareas en lenguaje natural
11. jat-project/jat-dataset-tokenized: versión tokenizada del jat-project/jat-dataset para aprendizaje por refuerzo
12. openai/gsm8k: Grade School Math 8K; 8.5K problemas de mates para búsqueda de respuestas
13. EssentialAI/essential-web-v1.0: 23.6B de documentos web clasificados según los códigos FDC
14. allenai/c4: versión limpia de Common Crawl idéntica a la versión Google de C4
15. nyu-mll/glue: benchmark General Language Understanding Evaluation; ver leaderboard
16. cais/mmlu: benchmark Measuring Massive Multitask Language Understanding; cuestiones de test de 57 tareas
17. bigcode/commitpackft: versión filtrada de bigcode/commitpack
18. TAUR-Lab/Taur_CoT_Analysis_Project___gpt-4o-mini-2024-07-18: dataset para analizar GPT-4o
19. CohereLabs/xP3x: Crosslingual Public Pool of Prompts eXtended; colección de prompts y datasets de Cohere Labs
20. m-a-p/PIN-100M: dataset multimodal de 14M muestras (150TB+, arXiv:2406.13923)
21. nicoboou/IDRCell100k: 104K imágenes microscópicas de células
22. openbmb/Ultra-FineWeb: análogo a HuggingFaceFW/fineweb
23. allenai/ai2_arc: 8K cuestiones de primaria sobre ciencia
24. HuggingFaceFW/fineweb-edu: análogo a HuggingFaceFW/fineweb
25. jat-project/jat-dataset: dataset para aprendizaje por refuerzo
26. laion/LAION-Audio-300M: dataset de audio anotado de LAION

3 Evaluate

Documentación de evaluate:

- **Get started:** evaluate
- **Tutoriales:** instalación, quicktour
- **Howtos:** elección de la métrica correcta, adición de nuevas evaluaciones, etc.
- **Guías conceptuales:** tipos de evaluaciones y consideraciones
- **Referencia:** clases principales, etc.

Ejemplo: con el dataset stanfordnlp/imdb; ver quicktour para más detalles

In [ ]:
```
from transformers import pipeline; from datasets import load_dataset
from evaluate import evaluator;  import evaluate
pipe = pipeline("text-classification", model="lvwerra/distilbert-imdb", device=0)
data = load_dataset("imdb", split="test").shuffle().select(range(1000))
metric = evaluate.load("accuracy")
```

```
Downloading builder script: 0.00B [00:00, ?B/s]
```

In [ ]:
```
task_evaluator = evaluator("text-classification")
results = task_evaluator.compute(model_or_pipeline=pipe, data=data, metric=metric,
                        label_mapping={"NEGATIVE": 0, "POSITIVE": 1},)
print(results)
```

```
{'accuracy': 0.929, 'total_time_in_seconds': 2.1861099790000935, 'samples_per_second': 457.4335278673357, 'latency_i
n_seconds': 0.0021861099790000935}
```

4 Búsqueda de respuestas

Tarea: **SQuAD2.0: The Stanford Question Answering Dataset**

- SQuAD1.1 contenía 100K+ pares pregunta-respuesta de 500+ artículos de Wikipedia
- SQuAD2.0 añade unas 50K preguntas similares a las de SQuAD1.1 pero imposibles de responder

Dataset: `rajpurkar/squad_v2` incluye 142192 preguntas; 130319 de training y 11873 de validación

In [ ]:
```
from datasets import load_dataset
ds = load_dataset("rajpurkar/squad_v2") # ver ds en HF
```

Métrica: squad_v2 devuelve

- `exact:` porcentaje de respuestas exactamente iguales a la referencia
- `f1:` score F1 promedio de los tokens predichos respecto a la referencia
- `total:` número de scores considerados (datos de test)
- `HasAns_exact:` `exact` para preguntas con respuesta
- `HasAns_f1:` `f1` para preguntas con respuesta
- `HasAns_total:` número de preguntas con respuesta
- `NoAns_exact:` `exact` para preguntas sin respuesta
- `NoAns_f1:` `f1` para preguntas sin respuesta
- `NoAns_total:` número de preguntas sin respuesta
- `best_exact:` `exact` con umbral variable
- `best_exact_thresh:` umbral de probabilidad de no-respuesta asociado a `best_exact`
- `best_f1:` `f1` con umbral variable
- `best_f1_thresh:` umbral de probabilidad de no-respuesta asociado a `best_f1`

Ejercicio: evalúa el modelo `timpal0l/mdeberta-v3-base-squad2` en validación con `squad_v2`

Solución:

In [ ]:
```python
from datasets import load_dataset
ds = load_dataset("rajpurkar/squad_v2", split="validation"); ds
```

Out[ ]:
```
Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 11873
})
```

In [ ]:
```python
from transformers import pipeline; import evaluate
task = "question-answering"; model = "timpal0l/mdeberta-v3-base-squad2"
pipe = pipeline(task=task, model=model, device_map="auto")
metric = evaluate.load("squad_v2")
evaluate.evaluator(task).compute(model_or_pipeline=pipe, data=ds, metric=metric, squad_v2_format=True)
```

Out[ ]:
```
{'exact': 79.81975911732502,
 'f1': 83.34595565756054,
 'total': 11873,
 'HasAns_exact': 77.32793522267207,
 'HasAns_f1': 84.39044054018491,
 'HasAns_total': 5928,
 'NoAns_exact': 82.3044575273339,
 'NoAns_f1': 82.3044575273339,
 'NoAns_total': 5945,
 'best_exact': 80.08085572306915,
 'best_exact_thresh': 1.9787678718566895,
 'best_f1': 83.62621976583806,
 'best_f1_thresh': 1.9787678718566895,
 'total_time_in_seconds': 194.91279194998788,
 'samples_per_second': 60.914421681705015,
 'latency_in_seconds': 0.01641647367556539}
```

5 Traducción automática

Dataset: Europarl-ST

- Dataset de traducción del habla multilíngüe extraído de debates del Parlamento Europeo entre 2008 y 2012
- Puede usarse para reconocimiento automático del habla (ASR), traducción automática (MT) y traducción del habla (ST)

Europarl-ST para MT en-X en HF: `tj-solergibert/Europarl-ST-processed-mt-en` consta de 770743 pares en-X; 602605 para training, 81968 para validación y 86170 de test; X $\in$ {de, fr, es, pl, pt, nl, it, ro}

In [ ]:
```
from datasets import load_dataset
ds = load_dataset("tj-solergibert/Europarl-ST-processed-mt-en"); ds
```

Out[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['source_text', 'dest_text', 'dest_lang'],
        num_rows: 602605
    })
    test: Dataset({
        features: ['source_text', 'dest_text', 'dest_lang'],
        num_rows: 86170
    })
    valid: Dataset({
        features: ['source_text', 'dest_text', 'dest_lang'],
        num_rows: 81968
    })
})
```

Métrica: sacrebleu devuelve `score` (score BLEU), `counts` (cuentas), `totals` (totales), `precisions` (precisiones), `bp` (penalización por brevedad), `sys_len` (longitud de las predicciones) y `ref_len` (longitud de la referencia)

Ejercicio: evalúa el modelo `facebook/nllb-200-distilled-600M` en test en-es con `sacrebleu`

Solución:

In [ ]:
```
from datasets import load_dataset
ds = load_dataset("tj-solergibert/Europarl-ST-processed-mt-en", split="test"); ds
```

Out[ ]:
```
Dataset({
    features: ['source_text', 'dest_text', 'dest_lang'],
    num_rows: 86170
})
```

In [ ]:
```
ds = ds.filter(lambda x: x["dest_lang"] == 2).remove_columns('dest_lang'); ds
```

Out[ ]:
```
Dataset({
    features: ['source_text', 'dest_text'],
    num_rows: 11008
})
```

In [ ]:
```
from transformers import pipeline; import evaluate
task = "translation"; model = "facebook/nllb-200-distilled-600M"
pipe = pipeline(task=task, model=model, device_map="auto", src_lang='eng', tgt_lang='spa_Latn')
metric = evaluate.load("sacrebleu")
evaluate.evaluator(task).compute(model_or_pipeline=pipe, data=ds, metric=metric, input_column='source_text',
    label_column='dest_text')
```

Out[ ]:
```
{'score': 44.82210843865245,
 'counts': [228068, 157506, 115455, 85516],
 'totals': [317671, 306663, 295655, 284655],
 'precisions': [71.79377406184386,
  51.36126627600982,
  39.0505826047251,
  30.041980643234794],
 'bp': 0.9828169445818209,
 'sys_len': 317671,
 'ref_len': 323177,
 'total_time_in_seconds': 2464.255756626997,
 'samples_per_second': 4.467068797707684,
 'latency_in_seconds': 0.2238604430075397}
```