Hoja informativa: Representaciones del lenguaje

Práctica

```
# clasificación mediante el uso de insertados
# df: dataset
# modelo: modelo de clasificación

import math
import numpy as np
import pandas as pd

import torch
import transformers

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

from tqdm.auto import tqdm

# comprobar la existencia de otros modelos en https://huggingface.co/transformers/pretrain
ed_models.html
tokenizer = transformers.BertTokenizer.from_pretrained('bert-base-uncased')
model = transformers.BertModel.from_pretrained('bert-base-uncased')

def BERT_text_to_embeddings(texts, max_length=512, batch_size=100, force_device=None, disa
ble_progress_bar=False):

    ids_list = []
    attention_mask_list = []

    # texto a los ID de relleno de tokens junto con sus máscaras de atención

    for input_text in tqdm(texts, disable=disable_progress_bar):
        ids = tokenizer.encode(input_text.lower(), add_special_tokens=True, truncation=Tru
e, max_length=max_length)
        padded = np.array(ids + [0]*(max_length - len(ids)))
        attention_mask = np.where(padded != 0, 1, 0)
        ids_list.append(padded)
        attention_mask_list.append(attention_mask)

    if force_device is not None:
        device = torch.device(force_device)
```

```
    else:
        device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    model.to(device)
    if not disable_progress_bar:
        print(f'Uso del dispositivo {device}.')

    # obtener insertados en lotes

    embeddings = []

    for i in tqdm(range(math.ceil(len(ids_list)/batch_size)), disable=disable_progress_ba
r):

        ids_batch = torch.LongTensor(ids_list[batch_size*i:batch_size*(i+1)]).to(device)
        attention_mask_batch = torch.LongTensor(attention_mask_list[batch_size*i:batch_siz
e*(i+1)]).to(device)

        with torch.no_grad():
            model.eval()
            batch_embeddings = model(input_ids=ids_batch, attention_mask=attention_mask_ba
tch)
        embeddings.append(batch_embeddings[0][:,0,:].detach().cpu().numpy())

    return np.concatenate(embeddings)

# ¡Atención!
# Ejecutar BERT para miles de textos puede llevar mucho tiempo en la CPU, al menos varias
 horas.
# Intenta encontrar una máquina con GPU que ejecute BERT en varios minutos en lugar de hor
as.
features = BERT_text_to_embeddings(df['text'])
target = df['target']

train_features, test_features, train_target, test_target = train_test_split(
    features, target, train_size=.8)

model.fit(train_features, train_target)

print(model.score(test_features, test_target))
```

Teoría

El insertado de palabras es un método de representación de textos con vectores respecto a un modelo lingüístico obtenido a partir de un gran corpus de textos.