Transformers

La arquitectura **transformer** es la más usada para resolver tareas de Procesamiento de Lenguaje Natural. Se basa en un **codificador** que recibe un texto de entrada y procesa la información para extraer características de cada una de las palabras en el texto. Puede producir vectores que tienen información semántica de las palabras, a estos vectores se les conoce como **encajes**.

En este notebook se obtendrán encajes para un texto usando modelos de lenguaje de huggingface.

Importante

Se recomienda usar una GPU para ejecutar este notebook. En el menú **Entorno de ejecución**, elegir Cambiar tipo de entorno de ejecución y en la sección **Acelerador por hardware** elegir una GPU o TPU.

```
from transformers import AutoTokenizer, AutoModelForMaskedLM, AutoModel
import torch
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
model_path = "dccuchile/bert-base-spanish-wwm-cased"
#model_path = "bertin-project/bertin-roberta-base-spanish"
#model_path = "google-bert/bert-base-multilingual-uncased"
```

```
device = "cuda:0" if torch.cuda.is_available() else "cpu"
print(device) ## Con esto verificamos que tengamos un dispositivo con cuda (GPU o TPU).
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModel.from_pretrained(model_path)
model = model.to(device)
```

```
cuda:0
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://hugg
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or
  warnings.warn(
```

```
tokenizer_config.json:   0%|          | 0.00/364 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/648 [00:00<?, ?B/s]
```

```
vocab.txt: 0.00B [00:00, ?B/s]
```

```
tokenizer.json: 0.00B [00:00, ?B/s]
```

```
special_tokens_map.json:   0%|          | 0.00/134 [00:00<?, ?B/s]
```

```
pytorch_model.bin:   0%|          | 0.00/440M [00:00<?, ?B/s]
```

```
Some weights of BertModel were not initialized from the model checkpoint at dccuchile/bert-ba
You should probably TRAIN this model on a down-stream task to be able to use it for predictio
```

Con este código se pueden obtener los encajes de una gran cantidad de textos.

```
splits = {'train': 'emo12-clean-corpus-mx-train.csv', 'test': 'emo12-clean-corpus-mx-test.csv
df = pd.read_csv("hf://datasets/guillermoruiz/MexEmojis/" + splits["train"])
```

```
model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]
```

```
df.head()
```

	Unnamed: 0	date	text	label	region
0	0	2022-05-01	Mi turbo carnala del alma _USR _emo en La Pl...		Mexico_City
1	1	2023-01-04	Las chicas de *Jimmy's* te esperan para pasar ...		Jalisco
2	2	2021-12-03	Big Time Rush y Mario Casas me quieren matar a...		BC

	Unnamed: 0	date	text	label	region
3	3	2022-05-03	Cosas de feria... Ayer _USR nos iba invitar lo...		Aguascalientes
4	4	2021-12-30	_USR Ya terminaste ....? _emo_emo_emo _URL		Jalisco

Usamos los primeros 20000 mensajes.

```
small_df = df[0:20000]
small_df.drop(columns=['Unnamed: 0', 'date', 'region'], inplace=True)
#small_df.to_csv("small_df.csv", index=False)
```

```
/tmp/ipython-input-2622146395.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  small_df.drop(columns=['Unnamed: 0', 'date', 'region'], inplace=True)
```

```
def predict(text, bs=128):
    output = []
    for i in range(0, len(text), bs):
        if i//bs%100==99:
            print(i, "/", len(text))
        tokens = tokenizer(text[i: i+bs], return_tensors="pt", padding='max_length', max_leng
        t = {"input_ids": tokens['input_ids'].to(device), "attention_mask": tokens['attentio

        with torch.no_grad():
            pred = model(**t).last_hidden_state[:,0].cpu()
        output.append(pred)
    output = torch.cat(output, dim=0)
    return output
```

```
output = predict(df['text'].to_list()[0:20000])
output.shape
```

```
12672 / 20000
```

```
torch.Size([20000, 768])
```

Se normalizan los vectores y se guardan en el archivo **encajes.npy**

```
output = torch.nn.functional.normalize(output, p=2, dim=1).numpy()
np.save("encajes_beto", output)
```