

MODULO_04 AJUSTE EMBEDDINGS CON TERMINOS TECNICOS

ATENCIÓN - FIJAR ESTAS VARIABLES ANTES DE EJECUTAR

```
In [15]: nombre_lote = "LOTE_20250614"
```

```
nombre_modulo = "MODULO_04"
```

```
In [17]: # -----  
# Configuración del entorno (Colab y Local)  
# -----  
  
try:  
    import google.colab  
    EN_COLAB = True  
except ImportError:  
    EN_COLAB = False  
  
if EN_COLAB:  
    from google.colab import drive  
    drive.mount("/content/drive", force_remount=True)  
    ruta_base = "/content/drive/MyDrive/TFM_EVA_MARTIN/Modulos"  
else:  
    ruta_base = "G:/Mi unidad/TFM_EVA_MARTIN/Modulos"  
  
print(f"Entorno detectado: {'Google Colab' if EN_COLAB else 'Local'}")  
print(f"Ruta base: {ruta_base}")  
  
lote_id = nombre_lote.replace("LOTE_", "")
```

Entorno detectado: Local

Ruta base: G:/Mi unidad/TFM_EVA_MARTIN/Modulos

```
In [18]: import sys  
import os  
ruta_config = os.path.join(ruta_base, "config.yaml")  
  
if ruta_base not in sys.path:  
    sys.path.append(ruta_base)  
import yaml  
  
# Cargar configuración desde el archivo YAML  
with open(ruta_config, "r", encoding="utf-8") as f:
```

```

config = yaml.safe_load(f)

config = yaml.safe_load(open(ruta_config))

# Extraer bloque de parámetros (KeyError si falta alguna clave)
params = config["parametros"]

```

Carga utilidades comunes e inicialización del entorno

```

In [19]: import pandas as pd

import utilidades_comunes

# 1. Configurar Logger
import logging
logger = utilidades_comunes.configurar_logger(nombre_modulo, ruta_logs=os.path.j
logger.setLevel(logging.DEBUG)

# 2. Inicializar entorno
entorno = utilidades_comunes.inicializar_entorno(nombre_modulo, nombre_lote, rut

```

```

2025-06-26 23:19:46,804 - INFO - 📁 Entorno inicializado para MODULO_04
2025-06-26 23:19:46,806 - INFO - 📁 Ruta entrada: G:/Mi unidad/TFM_EVA_MARTIN/Mo
dulos\MODULO_03\./salida
2025-06-26 23:19:46,809 - INFO - 📁 Ruta salida: G:/Mi unidad/TFM_EVA_MARTIN/Mod
ulos\MODULO_04\./salida
2025-06-26 23:19:46,813 - INFO - 📁 Ruta logs: G:/Mi unidad/TFM_EVA_MARTIN/Modul
os\MODULO_04\./logs
2025-06-26 23:19:46,818 - INFO - 📁 Ruta ejemplos: G:/Mi unidad/TFM_EVA_MARTIN/M
odulos\MODULO_04\./ejemplos
2025-06-26 23:19:46,821 - INFO - 🔗 Módulo anterior: MODULO_03
2025-06-26 23:19:46,823 - INFO - 🆔 Lote ID: 20250614

```

```

In [20]: # -----
# 2. Cargar términos técnicos desde JSON
# -----

import json
ruta_json = os.path.join(entorno["ruta_entrada"], "terminos_tecnicos.json")
with open(ruta_json, "r", encoding="utf-8") as f:
    terminos_tecnicos = json.load(f)

logger.info(f"📄 Términos técnicos cargados: {len(terminos_tecnicos)}")

```

```

2025-06-26 23:19:49,690 - INFO - 📄 Términos técnicos cargados: 14

```

Carga y analisis dataset de entrada

```

In [21]: patron_busqueda = os.path.join(
    entorno["ruta_entrada"],
    f"dataset_{entorno['nombre_modulo_anterior'].lower()}_{entorno['lote_id']}.*"
)

import glob
archivos_encontrados = glob.glob(patron_busqueda)

```

```
if not archivos_encontrados:
    raise FileNotFoundError(f"No se encontró archivo de entrada para el lote {no

fichero_entrada = archivos_encontrados[0]
df_entrada = utilidades_comunes.cargar_dataset(fichero_entrada, logger=logger)

utilidades_comunes.mostrar_muestra_dataset(df_entrada, "dataset de entrada", log
utilidades_comunes.guardar_muestra_dataset(df_entrada, "entrada", entorno["ruta_

df = df_entrada.copy()
```

[illegible]


```

6_procesado_limpio.txt | Positivo | Agente: Le atiende el contestador de 622931
01.
| Agente: Le atiende el contestador de 62293101.
|
| Neutro_SD23-04236_[965021512]_MERIDIANO_900_GEN._2023-03-10_20-55-21_691041377_
procesado_limpio.txt | Neutro | Agente: Bienvenido Meridiano. Por su seguri
dad, esta llamada podrá ser grabada. Usted consiente en que los datos que facilit
e se incorporen en un fichero titularidad de Meridiano S. A. con la finalidad de
gestionar la prestación del servicio. Si ya conoce nuestra política de protección
de datos y no desea escucharla nuevamente, pulse o diga cero. Si su llamada es pa
ra comunicar un fallecimiento, pulse o diga uno. Si desea comunicar un siniestro
de hogar, por favor, no se retire. En breve será atendido por uno de nuestros age
ntes. Por favor, no se retire. Por favor, no se | Agente: Bienvenido
Meridiano. Por su seguridad, esta llamada podrá ser grabada. Usted consiente en q
ue los datos que facilite se incorporen en un fichero titularidad de Meridiano S.
A. con la finalidad de gestionar la prestación del servicio. Si ya conoce nuestra
política de protección de datos y no desea escucharla nuevamente, pulse o diga ce
ro. Si su llamada es para comunicar un fallecimiento, pulse o diga uno. Si desea
comunicar un siniestro de hogar, por favor, no se retire. En breve será atendido
por uno de nuestros agentes. Por favor, no se retire. Por favor, no se
|
| Neutro_SD23-03776_[965021512]_MERIDIANO_900_GEN._2023-03-03_10-53-29_609484704_
procesado_limpio.txt | Neutro | Agente: de Asistencia a Seguros Meridiano.
Me conoce usted familiar, la nuera de DIFUNTO. ¿Me permite confirmar que ya está
en el domicilio, calle DIRECCION? Cliente: Sí. Agente: ¿El médico ya hubiera pasa
do por la casa? Cliente: No ha pasado todavía. Agente: ¿El servicio de asistencia
a seguros está en la casa? Cliente: No. Agente: ¿Le llamará directamente el compa
ñero que les vaya a atender presencialmente a la familia? Cliente: No. Agente: Le
voy a mandar también un mensaje al móvil con el teléfono de asistencia gratuita,
24 horas, por si alguna duda les pueda surgir a la familia. | Agente: de Asistenc
ia a Seguros Meridiano. Me conoce usted familiar, la nuera de DIFUNTO. ¿Me permit
e confirmar que ya está en el domicilio, calle DIRECCION? Cliente: Sí. Agente: ¿E
l médico ya hubiera pasado por la casa? Cliente: No ha pasado todavía. Agente: ¿E
l servicio de asistencia a seguros está en la casa? Cliente: No. Agente: ¿Le llam
ará directamente el compañero que les vaya a atender presencialmente a la famili
a? Cliente: No. Agente: Le voy a mandar también un mensaje al móvil con el teléfo
no de asistencia gratuita, 24 horas, por si alguna duda les pueda surgir a la fam
ilia. |
2025-06-26 23:19:53,321 - INFO -
--- Estadísticas básicas ---
| | count | unique | top
| freq |
|:-----|-----:|-----:|:-----
-----
-----
-----
-----
-----|-----:|
| nomfichero | 10 | 10 | Neutro_SD23-04271_[965021512]_MERIDIANO
_900_GEN._2023-03-12_11-30-47_658438667_procesado_limpio.txt
| 1 |
| etiqueta | 10 | 3 | Positivo
| 4 |
| transcripcion | 10 | 10 | Agente: Le atiende el contestador de 65
8. Comienza a transcribir la conversación: ¿Qué es lo que ha sucedido con su seg
uro de vida? ¿Ha recibido algún tipo de comunicación de nuestra parte? ¿Necesita
ayuda en algo en particular? ¿Puede decirme qué ha sucedido exactamente? ¿Ha habi
do algún problema con la cobertura? ¿Ha recibido algún tipo de comunicación de nu
estra parte | 1 |
| texto_etiquetado | 10 | 10 | Agente: Le atiende el contestador de 65

```

8. Comienza a transcribir la conversación: ¿Qué es lo que ha sucedido con su seguro de vida? ¿Ha recibido algún tipo de comunicación de nuestra parte? ¿Necesita ayuda en algo en particular? ¿Puede decirme qué ha sucedido exactamente? ¿Ha habido algún problema con la cobertura? ¿Ha recibido algún tipo de comunicación de nuestra parte | 1 |

2025-06-26 23:19:53,324 - INFO - -----

2025-06-26 23:19:53,374 - INFO -  Muestra guardada en G:/Mi unidad/TFM_EVA_MARTIN/Modulos\MODULO_04\./ejemplos\muestra_entrada.csv (5 filas)

```
In [25]: # -----
# 3. Preparación de textos para ajuste de embeddings (sin división si es pequeño)
# -----
from collections import Counter

# Limpieza y mapeo de etiquetas
etiquetas_map = {"Positivo": 0, "Neutro": 1, "Negativo": 2}
df = df.dropna(subset=["texto_etiquetado"])
df["texto_etiquetado"] = df["texto_etiquetado"].astype(str).str.strip()
df["label"] = df["etiqueta"].map(etiquetas_map)

# Comprobamos si se puede dividir de forma segura
conteo = Counter(df["label"])
minimo_por_clase = min(conteo.values())

if minimo_por_clase < 3 or len(df) < 30:
    logger.warning("⚠ Dataset demasiado pequeño para dividir. Usando todo como train_texts = df[\"texto_etiquetado\"].tolist()
else:
    from sklearn.model_selection import train_test_split

    df_train, df_temp = train_test_split(df, test_size=0.2, random_state=42, stratify=df["label"])
    df_val, df_test = train_test_split(df_temp, test_size=0.5, random_state=42, stratify=df["label"])

    train_texts = df_train["texto_etiquetado"].tolist()
    val_texts = df_val["texto_etiquetado"].tolist()
    test_texts = df_test["texto_etiquetado"].tolist()

    train_labels = df_train["label"].tolist()
    val_labels = df_val["label"].tolist()
    test_labels = df_test["label"].tolist()

    # Guardar subconjuntos
    df_train.to_csv(os.path.join(entorno["ruta_salida"], "train.csv"), index=False)
    df_val.to_csv(os.path.join(entorno["ruta_salida"], "val.csv"), index=False)
    df_test.to_csv(os.path.join(entorno["ruta_salida"], "test.csv"), index=False)

    logger.info(f"📁 Dataset dividido: {len(train_texts)} train, {len(val_texts)} val, {len(test_texts)} test")

# Mostrar conteo de clases
print("\n📊 Número de muestras por clase:")
print(df["etiqueta"].value_counts())

# Mostrar algunos ejemplos de transcripciones
print("\n📄 Ejemplo de transcripciones:")
for i, fila in df.sample(n=min(2, len(df))).iterrows(): # muestra 2 ejemplos como máximo
    print(f"\n📁 {fila['nomfichero']} [{fila['etiqueta']}]")
    print(f"{fila['texto_etiquetado'][:300]}...")
```

2025-06-26 23:36:04,160 - WARNING - ⚠ Dataset demasiado pequeño para dividir. Usando todo como train_texts.

📊 Número de muestras por clase:

etiqueta	
Positivo	4
Neutro	3
Negativo	3

Name: count, dtype: int64

📄 Ejemplo de transcripciones:

📁 Neutro_SD23-04236_[965021512]_MERIDIANO_900_GEN._2023-03-10_20-55-21_69104137_7_procesado_limpio.txt [Neutro]

Agente: Bienvenido Meridiano. Por su seguridad, esta llamada podrá ser grabada. Usted consiente en que los datos que facilite se incorporen en un fichero titularidad de Meridiano S. A. con la finalidad de gestionar la prestación del servicio. Si ya conoce nuestra política de protección de datos y no...

📁 Negativo_[965021512]_MERIDIANO_900_GEN._2023-03-04_15-34-14_procesado_limpio.txt [Negativo]

Cliente: XXXXXX. XXXXXX. Hola, buenas tardes. Doña XXXXXX, soy XXXXXX de la Central de Asistencia de Seguros Meridiano. XXXXXX. XXXXXX, hola, buenas tardes. XXXXX X. Hola, buenas tardes. Mire, de lo que hemos hablado usted y yo esta mañana, yo he procedido a informar a coordinación y tal. Y, bueno, h...

PASO 4: Procesamiento específico del módulo

```
In [26]: # — 4. Cargar modelo y tokenizador Longformer —
from transformers import LongformerTokenizerFast, LongformerForSequenceClassification
import torch

logger.info("🔗 Cargando modelo y tokenizador Longformer...")
tokenizer = LongformerTokenizerFast.from_pretrained("allenai/longformer-base-4096")
model = LongformerForSequenceClassification.from_pretrained("allenai/longformer-base-4096")

# Añadir nuevos tokens y redimensionar embeddings
tokenizer.add_tokens(terminos_tecnicos)
model.resize_token_embeddings(len(tokenizer))

# Congelar todo excepto embeddings
for param in model.parameters():
    param.requires_grad = False
model.longformer.embeddings.word_embeddings.weight.requires_grad = True
```

2025-06-26 23:36:35,356 - INFO - 🔗 Cargando modelo y tokenizador Longformer...
Some weights of LongformerForSequenceClassification were not initialized from the model checkpoint at allenai/longformer-base-4096 and are newly initialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias', 'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
In [ ]: # -----
# 5. Entrenamiento de Embeddings
# -----
from torch.utils.data import Dataset, DataLoader
```

```

# Si no hiciste división y no tienes train_labels, generamos etiquetas vacías o
if "train_labels" not in locals() or len(train_labels) != len(train_texts):
    logger.warning("⚠ No se encontraron etiquetas. Se asignarán etiquetas fict:
    train_labels = [0] * len(train_texts)

# Tokenizar con truncado y padding explícito
inputs = tokenizer(
    train_texts,
    truncation=True,
    padding="max_length",
    max_length=512,
    return_tensors="pt"
)

# Añadir etiquetas
inputs["labels"] = torch.tensor(train_labels)

class SimpleDataset(Dataset):
    def __init__(self, encodings):
        self.encodings = encodings

    def __getitem__(self, idx):
        return {key: val[idx] for key, val in self.encodings.items()}

    def __len__(self):
        return len(self.encodings["input_ids"])

# Crear dataset y dataloader
dataset = SimpleDataset(inputs)
train_dataloader = DataLoader(dataset, batch_size=1, shuffle=True)

# Optimizador solo para los embeddings entrenables
optimizer = torch.optim.AdamW(filter(lambda p: p.requires_grad, model.parameters))

logger.info("🚀 Iniciando ajuste de embeddings...")
model.train()
for epoch in range(2):
    for batch in train_dataloader:
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
        logger.info(f"Epoch {epoch} - Loss: {loss.item():.4f}")

```



```

2025-06-26 23:43:49,766 - WARNING - ⚠ No se encontraron etiquetas. Se asignarán
etiquetas dummy (0) a todo el dataset.
2025-06-26 23:43:49,974 - INFO - 🚀 Iniciando ajuste de embeddings...
2025-06-26 23:44:04,874 - INFO - Epoch 0 - Loss: 0.9894
2025-06-26 23:44:15,062 - INFO - Epoch 0 - Loss: 1.1406
2025-06-26 23:44:25,591 - INFO - Epoch 0 - Loss: 0.9994
2025-06-26 23:44:35,176 - INFO - Epoch 0 - Loss: 1.1419
2025-06-26 23:44:45,175 - INFO - Epoch 0 - Loss: 1.0194
2025-06-26 23:44:54,703 - INFO - Epoch 0 - Loss: 0.9772
2025-06-26 23:45:04,745 - INFO - Epoch 0 - Loss: 1.0798
2025-06-26 23:45:15,242 - INFO - Epoch 0 - Loss: 1.0413
2025-06-26 23:45:26,022 - INFO - Epoch 0 - Loss: 1.0314
2025-06-26 23:45:36,368 - INFO - Epoch 0 - Loss: 1.0121
2025-06-26 23:45:46,336 - INFO - Epoch 1 - Loss: 0.9759
2025-06-26 23:45:56,249 - INFO - Epoch 1 - Loss: 0.9383
2025-06-26 23:46:06,184 - INFO - Epoch 1 - Loss: 0.9970
2025-06-26 23:46:16,242 - INFO - Epoch 1 - Loss: 0.9832
2025-06-26 23:46:25,659 - INFO - Epoch 1 - Loss: 1.0032
2025-06-26 23:46:35,445 - INFO - Epoch 1 - Loss: 0.9994
2025-06-26 23:46:44,732 - INFO - Epoch 1 - Loss: 1.0748
2025-06-26 23:46:54,765 - INFO - Epoch 1 - Loss: 1.0181
2025-06-26 23:47:04,660 - INFO - Epoch 1 - Loss: 1.0416
2025-06-26 23:47:14,011 - INFO - Epoch 1 - Loss: 1.0868

```

```

In [29]: # -----
# 6. Guardar modelo ajustado
# -----
modelo_dir = os.path.join(entorno["ruta_salida"], "modelo_ajustado")
os.makedirs(modelo_dir, exist_ok=True)
model.save_pretrained(modelo_dir)
tokenizer.save_pretrained(modelo_dir)
logger.info(f"📁 Modelo y tokenizador ajustados guardados en: {modelo_dir}")

logger.info(f"✅ Finalización del procesamiento del módulo {nombre_modulo}")

```

```

2025-06-26 23:52:56,418 - INFO - 📁 Modelo y tokenizador ajustados guardados en:
G:/Mi unidad/TFM_EVA_MARTIN/Modulos/MODULO_04\./salida\modelo_ajustado
2025-06-26 23:52:56,420 - INFO - ✅ Finalización del procesamiento del módulo MO
DULO_04
2025-06-26 23:52:56,420 - INFO - ✅ Finalización del procesamiento del módulo MO
DULO_04

```

```

In [30]: print("📖 Tokens añadidos al vocabulario:")
print(terminos_tecnicos[:10]) # primeros 10

print(f"\n📊 Tamaño total del vocabulario tras expansión: {len(tokenizer)}")

# Mostrar ejemplo de cómo el modelo tokeniza una frase con términos nuevos
frase = "La tanatopraxia fue solicitada para el velatorio en Olvera."
tokens = tokenizer.tokenize(frase)
ids = tokenizer.convert_tokens_to_ids(tokens)

print("\n🔍 Tokenización de ejemplo:")
for tok, idx in zip(tokens, ids):
    print(f"{tok:<20} → id {idx}")

```

📄 Tokens añadidos al vocabulario:
['doña', 'cliente', 'datos', 'fallecimiento', 'gracias', 'hospital', 'momentito', 'propiedad', 'resto', 'doña cliente']

📊 Tamaño total del vocabulario tras expansión: 50278

🔍 Tokenización de ejemplo:

La	→ id 10766
Ĝtan	→ id 15149
at	→ id 415
op	→ id 1517
rax	→ id 35054
ia	→ id 493
Ĝfue	→ id 13081
Ĝsolicit	→ id 22706
ada	→ id 2095
Ĝpara	→ id 3840
Ĝel	→ id 1615
Ĝvel	→ id 23021
ator	→ id 2630
io	→ id 1020
Ĝen	→ id 1177
Ĝol	→ id 5387
ver	→ id 2802
a	→ id 102
.	→ id 4