03_embeddings_estaticos

October 27, 2025

[1]:
```
# Crear embeddings estáticos desde processed.
# Modelo: Word2Vec entrenado en TRAIN.
# Salidas: modelo, vocab, sentence-embeddings por split y cobertura OOV.
```

Imports y config

[2]:
```
from pathlib import Path
import json
import numpy as np
import pandas as pd

from gensim.models import Word2Vec
from gensim.models.keyedvectors import KeyedVectors

SEED = 42
DIM = 128
EPOCHS = 8
MIN_COUNT = 5
WINDOW = 5
SG = 1            # 1 skip-gram, 0 CBOW

np.random.seed(SEED)
pd.set_option("display.max_colwidth", 120)
```

Rutas

[3]:
```
def find_root():
    p = Path.cwd()
    for cand in [p, *p.parents]:
        if (cand / "data" / "processed").exists():
            return cand
    raise FileNotFoundError("No encuentro data/processed.")

ROOT = find_root()
PROC = ROOT / "data" / "processed"
FEAT = ROOT / "features" / "embeddings_static"
FEAT.mkdir(parents=True, exist_ok=True)

NIVELES = ["easy","medium","hard"]
```

```
SPLITS = ["train","validation"]
```

Carga de processed → tokens

[4]:
```
def cargar_split(split):
    rows = []
    for level in NIVELES:
        p = PROC / level / split / "sentences.jsonl"
        if not p.exists():
            continue
        df = pd.read_json(p, lines=True, dtype={"sent_id": int})
        df["level"] = level
        df["split"] = split
        rows.append(df[["level","split","doc_id","sent_id","text_norm"]])
    if not rows:
        return pd.
 ↪DataFrame(columns=["level","split","doc_id","sent_id","text_norm"])
    out = pd.concat(rows, ignore_index=True).
 ↪sort_values(["level","doc_id","sent_id"])
    return out.reset_index(drop=True)

df_train = cargar_split("train")
df_val   = cargar_split("validation")
print(df_train.shape, df_val.shape)

def texto_a_tokens(s):
    # text_norm ya está en minúsculas y normalizado
    return str(s).split()

sentences_train = [texto_a_tokens(t) for t in df_train["text_norm"]]
```

```
(171602, 5) (36558, 5)
```

Entrenamiento Word2Vec en train

[5]:
```
w2v = Word2Vec(
    sentences=sentences_train,
    vector_size=DIM,
    window=WINDOW,
    min_count=MIN_COUNT,
    sg=SG,
    negative=10,
    epochs=EPOCHS,
    workers=4,
    seed=SEED
)
kv: KeyedVectors = w2v.wv
print("Vocab size:", len(kv.key_to_index))
```

```
Vocab size: 17403
```

Guardado del modelo y vocab

[6]:
```
# Solo KeyedVectors para aligerar
kv.save(str(FEAT / "w2v.kv"))

# Vocabulario con frecuencias
vocab_rows = []
for w, idx in kv.key_to_index.items():
    cnt = kv.get_vecattr(w, "count") if kv.has_index_for(w) else None
    vocab_rows.append((w, idx, cnt))
pd.DataFrame(vocab_rows, columns=["token","index","count"]).to_csv(FEAT /␣
 ↪"w2v_vocab.csv", index=False)
```

Embeddings de frase train y validation

[7]:
```
def oracion_a_vector(tokens, kv, dim=DIM):
    vecs = [kv[t] for t in tokens if t in kv]
    if not vecs:
        return np.zeros(dim, dtype=np.float32)
    v = np.mean(vecs, axis=0)
    return v.astype(np.float32)

def construir_sentence_embeddings(df, kv, split):
    X = np.zeros((len(df), kv.vector_size), dtype=np.float32)
    seen = 0
    for i, toks in enumerate(map(texto_a_tokens, df["text_norm"])):
        vec = oracion_a_vector(toks, kv)
        if vec.any():
            seen += 1
        X[i] = vec
    idx = df[["level","split","doc_id","sent_id"]].reset_index(drop=True)
    np.save(FEAT / f"S_{split}.npy", X)
    idx.to_csv(FEAT / f"S_{split}_index.csv", index=False)
    return X, idx, seen

Xtr, idx_tr, seen_tr = construir_sentence_embeddings(df_train, kv, "train")
Xva, idx_va, seen_va = construir_sentence_embeddings(df_val,   kv, "validation")

print("Sentence embeddings:")
print("train:", Xtr.shape, "cobertura oraciones con al menos 1 token conocido:
 ↪", round(seen_tr/len(df_train), 3))
print("val  :", Xva.shape, "cobertura oraciones con al menos 1 token conocido:
 ↪", round(seen_va/len(df_val), 3))
```

```
Sentence embeddings:
train: (171602, 128) cobertura oraciones con al menos 1 token conocido: 0.987
val  : (36558, 128) cobertura oraciones con al menos 1 token conocido: 0.988
```

Cobertura OOV y resumen

[8]:
```
def cobertura_tokens(df, kv):
    total = 0
    hit = 0
    for toks in map(texto_a_tokens, df["text_norm"]):
        total += len(toks)
        hit += sum(1 for t in toks if t in kv)
    rate = hit / total if total else 0.0
    return {"tokens_total": int(total), "tokens_conocidos": int(hit),␣
 ↪"coverage": float(round(rate, 4))}

cov = {
    "train": cobertura_tokens(df_train, kv),
    "validation": cobertura_tokens(df_val, kv),
    "vocab_size": len(kv.key_to_index),
    "dim": kv.vector_size,
    "min_count": MIN_COUNT,
    "window": WINDOW,
    "epochs": EPOCHS,
    "sg": SG
}
(Path(FEAT) / "w2v_resumen.json").write_text(json.dumps(cov, indent=2),␣
 ↪encoding="utf-8")
cov
```

[8]:
```
{'train': {'tokens_total': 3112716,
  'tokens_conocidos': 3064989,
  'coverage': 0.9847},
 'validation': {'tokens_total': 668938,
  'tokens_conocidos': 655930,
  'coverage': 0.9806},
 'vocab_size': 17403,
 'dim': 128,
 'min_count': 5,
 'window': 5,
 'epochs': 8,
 'sg': 1}
```

Exportables

[9]:
```
print("Guardado:")
print("-", FEAT / "w2v.kv")
print("-", FEAT / "w2v_vocab.csv")
print("-", FEAT / "S_train.npy", "y", FEAT / "S_train_index.csv")
print("-", FEAT / "S_validation.npy", "y", FEAT / "S_validation_index.csv")
print("-", FEAT / "w2v_resumen.json")
```

```
Guardado:
- /Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
```

analysis/features/embeddings_static/w2v.kv
- /Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/w2v_vocab.csv
- /Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/S_train.npy y
/Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/S_train_index.csv
- /Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/S_validation.npy y
/Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/S_validation_index.csv
- /Users/eeguskiza/Documents/Deusto/2025/NLP/multi-author-
analysis/features/embeddings_static/w2v_resumen.json