

# MODULO\_03 EXTRACCIÓN LISTA DE TERMINOS RELEVANTES

---

## ATENCIÓN - FIJAR ESTAS VARIABLES ANTES DE EJECUTAR

---

```
In [21]: nombre_lote = "LOTE_20250614"
```

```
nombre_modulo = "MODULO_03"
```

```
In [22]: # -----  
# 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 [23]: 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 [24]: 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

```

```

2025-06-24 00:16:56,866 - INFO -  Entorno inicializado para MODULO_03
2025-06-24 00:16:56,869 - INFO -  Ruta entrada: G:/Mi unidad/TFM_EVA_MARTIN/Mod
ulos\MODULO_02\./salida
2025-06-24 00:16:56,871 - INFO -  Ruta salida: G:/Mi unidad/TFM_EVA_MARTIN/Mod
ulos\MODULO_03\./salida
2025-06-24 00:16:56,874 - INFO -  Ruta logs: G:/Mi unidad/TFM_EVA_MARTIN/Modul
os\MODULO_03\./logs
2025-06-24 00:16:56,878 - INFO -  Ruta ejemplos: G:/Mi unidad/TFM_EVA_MARTIN/M
odulos\MODULO_03\./ejemplos
2025-06-24 00:16:56,880 - INFO -  Módulo anterior: MODULO_02
2025-06-24 00:16:56,883 - INFO -  Lote ID: 20250614

```

## Carga y analisis dataset de entrada

```

In [25]: 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()

```

```
2025-06-24 00:17:01,129 - INFO -  Dataset cargado desde G:/Mi unidad/TFM_EVA_MARTIN\Modulos\MODULO_02\./salida\dataset_modulo_02_20250614.csv (4 filas, 3 columnas)
2025-06-24 00:17:01,130 - INFO - --- Muestra de dataset de entrada (primeras 5 filas) ---
2025-06-24 00:17:01,134 - INFO - Filas totales: 4, Columnas totales: 3
```



| Negativo\_SD23-03822\_[965021512]\_MERIDIANO\_900\_GEN.\_2023-03-03\_23-08-14\_620135618\_procesado\_limpio.txt | Negativo | Agente: Buenas noches, doña CLIENTE. Mi nombre es AGENTE, le llamo del Departamento de Asistencia de Seguros Meridiano. Doña CLIENTE, en primer lugar, trasladarle tanto a usted como al resto de familiares nuestras más sinceras condolencias por el fallecimiento de doña DIFUNTO, de su madre. Cliente: De nada, doña CLIENTE. Agente: Doña CLIENTE, le llamo porque nos ha informado Isabel, la cobradora, nos ha indicado que su madre se encuentra en el hospital de Ronda, ¿es correcto? Cliente: Sí, es correcto. Agente: Y desean ustedes realizar el velatorio en Olvera, ¿es así? Cliente: Sí, sí. Agente: En calle Azuaga 14, en el territorio, no. A ver, un momentito, en calle Azuaga me dijo. Cliente: Calle Azuaga 14. Agente: Calle Azuaga 14, vale. Posteriormente, ¿lo que desearían dejó doña DIFUNTO indicado si lo que ella quería era entierro o incineración? Cliente: No, entierro, entierro. Agente: Entierro, vale. ¿Tienen ustedes dicho en propiedad, doña CLIENTE? Cliente: Sí, sí, tiene, tiene. Agente: Dicho en propiedad, vale. Pues un momentito que lo note aquí. Cliente: Doña CLIENTE, el dicho que tienen ustedes en propiedad, ¿saben si está ocupado o está vacío? Agente: Está ocupado con resto, pero vamos, están los restos en saco y todo. O sea, que ya hace muchos años, ¿verdad? Cliente: Sí, muchísimos. Agente: Vale. Muchísimos, son de los padres. Ah, vale. Ella ha muerto con 94 años. Vale. Agente: Pues, doña DIFUNTO, o perdón, doña CLIENTE, le comento, voy a dar ya mismo parte a los servicios funerarios para que ellos vayan a atenderla al hospital. Cliente: Vale. Agente: ¿Llaman a ellos para indicarle cuánto tiempo aproximadamente van a tardar en llegar? Cliente: Vale. Agente: Nosotros, doña CLIENTE, quedamos a su disposición en un teléfono gratuito que va a recibir a través de un mensaje y estamos las 24 horas, por si nos necesita para algo. Cliente: Vale. Agente: Doña CLIENTE, bueno, entiendo que el fallecimiento de doña DIFUNTO no guarda ninguna relación con el COVID-19, ¿es así? Cliente: No, no, no, nada. Agente: Vale. Otras cosas, otras cuestiones. Vale. Agente: Pues cualquier cosita, como le indicaba, quedamos a su disposición y nuevamente nuestras condolencias por la pérdida. Cliente: Vale. Agente: Entonces, ¿a ellos nos llaman y no avisan? Cliente: Sí, yo ahora voy a llamarles, les voy a dar todos los datos y lo que ustedes desean hacer y su número de teléfono para que ellos le llamen. Agente: ¿De acuerdo? Cliente: Vale, vale. Agente: Muchas gracias. A usted, y de verdad que lamentamos la pérdida. Cliente: Gracias. Agente: Un saludo. Cliente: Gracias. Agente: Gracias. Cliente: Muchas gracias. |

2025-06-24 00:17:01,183 - INFO -

--- Estadísticas básicas ---

	count	unique	top
freq			

1			
---	--	--	--

2			
---	--	--	--

3			
---	--	--	--

4			
---	--	--	--

5			
---	--	--	--

6			
---	--	--	--

7			
---	--	--	--

8			
---	--	--	--

9			
---	--	--	--

10			
----	--	--	--

11			
----	--	--	--

12			
----	--	--	--

13			
----	--	--	--

14			
----	--	--	--

15			
----	--	--	--

16			
----	--	--	--

17			
----	--	--	--

18			
----	--	--	--

19			
----	--	--	--

20			
----	--	--	--


21			
----	--	--	--

22			
----	--	--	--

23			
----	--	--	--

24			
----	--	--	--

25			
----	--	--	--

2025-06-24 00:17:01,322 - INFO -  Muestra guardada en G:/Mi unidad/TFM\_EVA\_MAR TIN/Modulos\MODULO\_03\./ejemplos\muestra\_entrada.csv (4 filas)

## PASO 4: Procesamiento específico del módulo

```
In [26]: import os
os.environ["USE_TF"] = "0"

import spacy
nlp = spacy.load("es_core_news_sm")

from sklearn.feature_extraction.text import TfidfVectorizer
import json
```

```
In [27]: def paso_1_preprocesado(df):
# Quedarnos solo con la columna 'texto' y eliminar filas vacías
df = df[['texto_etiquetado']].dropna()

# Convertir a minúsculas y quitar espacios al inicio/final
df['texto_etiquetado'] = df['texto_etiquetado'].str.lower().str.strip()

# eliminar espacios múltiples intermedios
df['texto_etiquetado'] = df['texto_etiquetado'].str.replace(r'\s+', ' ', reg

return df
```

```
In [28]: def paso_2_extraccion(df, nlp):
"""
Extrae frases nominales y entidades nombradas de df['texto'] usando spaCy.
Devuelve una lista de términos limpios y únicos.
"""
docs = nlp.pipe(df['texto_etiquetado'].astype(str).tolist(), batch_size=32)

frases = []
for doc in docs:
    frases.extend([chunk.text.strip() for chunk in doc.noun_chunks])
    frases.extend([ent.text.strip() for ent in doc.ents])

# Normalizar y deduplicar manteniendo orden
frases = [f.lower() for f in frases if f]
return list(dict.fromkeys(frases))
```

```
In [29]: def procesamiento(df):
# Paso 1: preprocesado
df_limpio = paso_1_preprocesado(df)
# Paso 2: extracción de frases y entidades
terminos = paso_2_extraccion(df_limpio, nlp)

# Aquí términos es la lista de sintagmas y entidades únicas, lista para pasa
return terminos
```

```
In [30]: salida = procesamiento(df)
print(salida[:10]) # muestra los primeros 10 términos
```

```
['agente: bienvenido meridiano', 'su seguridad', 'esta llamada', 'usted', 'los da
tos', 'que', 'un fichero titularidad', 's.', 'la finalidad', 'la prestación']
```

```
In [31]: from spacy.lang.es.stop_words import STOP_WORDS as stop_words_es
from sklearn.feature_extraction.text import TfidfVectorizer

# Term Frequency - Inverse Document Frequency, mide la importancia de un término

def paso_3_tfidf(frases, max_features=50, ngram_range=(1,2)):
    vectorizer = TfidfVectorizer(
        max_features=max_features,
        stop_words=list(stop_words_es),
        ngram_range=ngram_range
    )
    X = vectorizer.fit_transform(frases)
    scores = X.sum(axis=0).A1
    terminos_scores = list(zip(vectorizer.get_feature_names_out(), scores))
    terminos_scores.sort(key=lambda x: x[1], reverse=True)
    return terminos_scores
```

```
In [32]: salida = procesamiento(df) # lista de sintagmas + entidades
print(salida[:10])
```

```
['agente: bienvenido meridiano', 'su seguridad', 'esta llamada', 'usted', 'los da
tos', 'que', 'un fichero titularidad', 's.', 'la finalidad', 'la prestación']
```

```
In [ ]: #usamos TF-IDF como mecanismo para puntuar y ordenar automáticamente los término
# (en nuestro caso, sintagmas nominales y entidades)
# extraídos de las transcripciones.
# Llamamos al TF-IDF para quedarnos con, p.ej., los 100 términos más relevantes:
tfidf_scores = paso_3_tfidf(salida, max_features=100)

# Veamos los 20 primeros con su score:
import pandas as pd
df_tfidf = pd.DataFrame(tfidf_scores, columns=["termino", "score"])
print(df_tfidf.head(20))
```

```
In [34]: from spacy.lang.es.stop_words import STOP_WORDS as stop_spacy
from stop_words import get_stop_words
import re

# 1) Lista combinada de stop-words
stop_sw = set(stop_spacy) | set(get_stop_words('spanish'))
print(f"Total stop-words combinadas: {len(stop_sw)}")
print("Algunas stop-words de ejemplo:", list(stop_sw)[:20])

# 2) Filtrado con la lista generada
def paso_4_filtrado(terminos_scores, stop_es):
    filtrados = []
    for t, s in terminos_scores:
        clave = re.sub(r'\W+', '', t).lower()
        if len(clave) < 3 or clave.isdigit() or clave in stop_es:
            continue
        filtrados.append((t, s))
    return filtrados

# Reaplicamos el filtrado a los tfidf_scores
filtrados = paso_4_filtrado(tfidf_scores, stop_sw)
df_filtrado = pd.DataFrame(filtrados, columns=["termino_tecnico", "score"])
```

```
print("\nPrimeros 20 términos tras aplicar stop-words combinadas:")
print(df_filtrado.head(20))
```

Total stop-words combinadas: 662

Algunas stop-words de ejemplo: ['cuatro', 'otras', 'nuestras', 'próximo', 'hubieran', 'todas', 'tuviésemos', 'nuevo', 'esas', 'puede', 'estuvieses', 'pueden', 'nunca', 'eran', 'hoy', 'e', 'estuvo', 'hubiesen', 'grande', 'tuviese']

Primeros 20 términos tras aplicar stop-words combinadas:

	termino_tecnico	score
0	doña	4.304155
1	cliente	3.710336
2	datos	3.000000
3	fallecimiento	2.000000
4	gracias	2.000000
5	hospital	2.000000
6	llamada	2.000000
7	momentito	2.000000
8	propiedad	2.000000
9	resto	2.000000
10	doña cliente	1.888170
11	difunto	1.858166
12	años	1.544838
13	calle	1.544838
14	teléfono	1.544838
15	madre	1.462176
16	agente	1.424358
17	condolencias	1.089676
18	agentes	1.000000
19	asistencia	1.000000

```
In [35]: def paso_4_pos_filter(terminos_scores, nlp, stop_es, min_score=1.5):
        """
        - Sólo mantenemos sintagmas cuya raíz sea NOUN o PROPN
        - Aplicamos la lista de stop-words combinada
        - Podemos además forzar un umbral mínimo de score TF-IDF
        """
        filtrados = []
        for t, s in terminos_scores:
            if s < min_score:
                continue
            # Volvemos a analizar el sintagma como Doc
            doc = nlp(t)
            # Convertimos todo el Doc en un Span para poder usar .root
            span = doc[:]
            head = span.root # ahora sí es un Token
            # 1) POS check
            if head.pos_ not in ("NOUN", "PROPN"):
                continue
            # 2) Stop-words sobre el Lema
            clave = head.lemma_.lower()
            if clave in stop_es:
                continue
            # 3) Longitud mínima del Lema
            if len(clave) < 3:
                continue
            filtrados.append((t, s))
        return filtrados
```



```
In [36]: filtrados = paso_4_pos_filter(tfidf_scores, nlp, stop_sw, min_score=1.5)
import pandas as pd
df_filtrado = pd.DataFrame(filtrados, columns=["termino_tecnico", "score"])
print(df_filtrado.head(20))
```

	termino_tecnico	score
0	doña	4.304155
1	cliente	3.710336
2	datos	3.000000
3	fallecimiento	2.000000
4	gracias	2.000000
5	hospital	2.000000
6	momentito	2.000000
7	propiedad	2.000000
8	resto	2.000000
9	doña cliente	1.888170
10	difunto	1.858166
11	años	1.544838
12	calle	1.544838
13	teléfono	1.544838

```
In [37]: # — 5. Guardar JSON con Lista de términos técnicos
try:
    # Usa entorno["ruta_salida"] como carpeta de salida estándar
    ruta_json = os.path.join(entorno["ruta_salida"], "terminos_tecnicos.json")

    # 1) Carga existentes (si los hay)
    if os.path.exists(ruta_json):
        with open(ruta_json, 'r', encoding='utf-8') as f:
            existentes = json.load(f)
    else:
        existentes = []

    # 2) Extrae la lista nueva de términos (filtrados puede venir de tu último p
    nuevos = [t for t, s in filtrados]

    # 3) Combina manteniendo orden y sin duplicados
    todos = list(dict.fromkeys(existentes + nuevos))

    # 4) Escribe de nuevo el JSON "acumulado"
    with open(ruta_json, 'w', encoding='utf-8') as f:
        json.dump(todos, f, ensure_ascii=False, indent=4)

    logger.info(f" Términos técnicos acumulados guardados: {len(todos)} entradas
    # 5) Cargar el JSON acumulado en un DataFrame df_salida
    #import pandas as pd
    #df_salida = pd.DataFrame(todos, columns=["termino_tecnico"])
    #logger.info(f"DataFrame df_salida creado con {len(df_salida)} términos")


except Exception as e_json:
    logger.error(f"Error al guardar o cargar términos técnicos: {e_json}", exc_i
```

```
2025-06-24 00:17:46,255 - INFO - Términos técnicos acumulados guardados: 14 entra
das en G:/Mi unidad/TFM_EVA_MARTIN/Modulos/MODULO_03/./salida/terminos_tecnicos.j
son
```

```
In [ ]: # -----
# PASO 5: Guardar dataset salida con nombre estándar
# guardamos el dataset que ya generamos en el paso anterior
# lo arrastramos al siguiente módulo
```



```
# -----
nombre_salida = os.path.join(
    entorno["ruta_salida"],
    f"dataset_{nombre_modulo.lower()}_{entorno['lote_id']}.csv"
)

utilidades_comunes.guardar_dataset(df_entrada, nombre_salida, logger=logger)
```

2025-06-24 00:17:51,359 - INFO -  Dataset guardado en: G:/Mi unidad/TFM\_EVA\_MARTIN/Modulos\MODULO\_03\./salida\dataset\_modulo\_03\_20250614.csv (4 filas, 3 columnas)

In [39]:

```
# -----
# PASO 6: Mostrar muestra final
# -----
nombre_muestra = f"{nombre_modulo.lower()}_{entorno['lote_id']}"
utilidades_comunes.mostrar_muestra_dataset(df_salida, nombre_muestra, logger=logger)

logger.info(f" Finalización del procesamiento del módulo {nombre_modulo}")
logger.info(f" Dataset final disponible en: {nombre_salida}")
```

2025-06-24 00:17:54,509 - INFO - --- Muestra de modulo\_03\_20250614 (primeras 5 filas) ---

2025-06-24 00:17:54,513 - INFO - Filas totales: 14, Columnas totales: 1

2025-06-24 00:17:54,564 - INFO -

termino_tecnico	
:-----	
doña	
cliente	
datos	
fallecimiento	
gracias	


2025-06-24 00:17:54,615 - INFO -


--- Estadísticas básicas ---

	count	unique	top	freq
:-----	:-----	:-----	:-----	:-----
termino_tecnico	14	14	doña	1

2025-06-24 00:17:54,720 - INFO - -----

-----

2025-06-24 00:17:54,780 - INFO -  Finalización del procesamiento del módulo MODULO\_03

2025-06-24 00:17:54,783 - INFO -  Dataset final disponible en: G:/Mi unidad/TFM\_EVA\_MARTIN/Modulos\MODULO\_03\./salida\dataset\_modulo\_03\_20250614.csv