NLP tópicos- dataset News Category HuffPost

May 20, 2026

1 Procesamiento y detección de tópicos con LDA

En este notebook se trabaja con el News Category Dataset de HuffPost.

El objetivo es construir un flujo completo de análisis de tópicos:

- Cargar un corpus de noticias,
- Explorar sus categorías,
- Construir un texto de trabajo,
- Aplicar un preprocesamiento básico,
- Representar los documentos mediante una matriz documento-término con conteos,
- Entrenar un modelo LDA,
- Interpretar los tópicos aprendidos,
- Analizar la mezcla de tópicos por documento,
- Comparar los resultados con las categorías reales del dataset,
- Probar distintos números de tópicos,
- Revisar algunas limitaciones,
- Añadir extensiones opcionales con lematización y TF-IDF.

1.1 1. Carga del dataset

El dataset, que se descarga automáticamente en el siguiente bloque de código, está representado formato JSON por líneas. Esto significa que cada línea del archivo contiene una noticia en formato JSON.

[1]:
```
!wget https://webs.um.es/slopez/datasets/BigData/News_Category_Dataset_HuffPost.
↪json
```

```
--2026-05-20 11:54:07--
https://webs.um.es/slopez/datasets/BigData/News_Category_Dataset_HuffPost.json
Resolving webs.um.es (webs.um.es)… 155.54.212.112
Connecting to webs.um.es (webs.um.es)|155.54.212.112|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 87295572 (83M) [application/json]
Saving to: ‘News_Category_Dataset_HuffPost.json’

News_Category_Datas 100%[===================>]  83.25M  84.1MB/s    in 1.0s

2026-05-20 11:54:08 (84.1 MB/s) - ‘News_Category_Dataset_HuffPost.json’ saved
```

[87295572/87295572]

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
from collections import Counter

# El dataset de HuffPost está representado en formato JSON por líneas.
df = pd.read_json('News_Category_Dataset_HuffPost.json', lines=True)

print("Dimensiones del dataset:", df.shape)
df.head()
```

```
Dimensiones del dataset: (209527, 6)
```

[2]:

	link	headline	category	short_description	authors	date
0	https://www.huffpost.com/entry/covid-boosters-…	Over 4 Million Americans Roll Up Sleeves For O…	U.S. NEWS	Health experts said it is too early to predict…	Carla K. Johnson, AP	2022-09-23
1	https://www.huffpost.com/entry/american-airlin…	American Airlines Flyer Charged, Banned For Li…	U.S. NEWS	He was subdued by passengers and crew when he …	Mary Papenfuss	2022-09-23
2	https://www.huffpost.com/entry/funniest-tweets…	23 Of The Funniest Tweets About Cats And Dogs …	COMEDY	"Until you have a dog you don't understand wha…	Elyse Wanshel	2022-09-23
3	https://www.huffpost.com/entry/funniest-parent…	The Funniest Tweets From Parents This Week (Se…	PARENTING	"Accidentally put grown-up toothpaste on my to…	Caroline Bologna	2022-09-23
4	https://www.huffpost.com/entry/amy-cooper-lose…	Woman Who Called Cops On Black Bird-Watcher Lo…	U.S. NEWS	Amy Cooper accused investment firm Franklin Te…	Nina Golgowski	2022-09-22

1.1.1 Inspección inicial

Antes de procesar el texto, revisamos qué columnas contiene el dataset.

- `link`: enlace,
- `headline`: titular,
- `category`: categoría original de la noticia,
- `short_description`: breve descripción,
- `authors`: autores,
- `date`: fecha.

Para el análisis de tópicos no usaremos la categoría como etiqueta de entrenamiento. La categoría se utilizará más adelante solo como ayuda para interpretar los resultados.

[3]:
```
df.info()
```

```
<class 'pandas.DataFrame'>
RangeIndex: 209527 entries, 0 to 209526
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   link               209527 non-null  str
 1   headline           209527 non-null  str
 2   category           209527 non-null  str
 3   short_description  209527 non-null  str
 4   authors            209527 non-null  str
 5   date               209527 non-null  datetime64[us]
dtypes: datetime64[us](1), str(5)
memory usage: 9.6 MB
```

[4]:
```
print("Ejemplo de noticia:")
print("- Link: ", df.iloc[0]["link"])
print("- Headline: ", df.iloc[0]["headline"])
print("- Category: ", df.iloc[0]["category"])
print("- Short description: ", df.iloc[0]["short_description"])
print("- Authors: ", df.iloc[0]["authors"])
print("- Date: ", df.iloc[0]["date"])
```

```
Ejemplo de noticia:
- Link:  https://www.huffpost.com/entry/covid-boosters-uptake-
us_n_632d719ee4b087fae6feaac9
- Headline:  Over 4 Million Americans Roll Up Sleeves For Omicron-Targeted COVID
Boosters
- Category:  U.S. NEWS
- Short description:  Health experts said it is too early to predict whether
demand would match up with the 171 million doses of the new boosters the U.S.
ordered for the fall.
- Authors:  Carla K. Johnson, AP
- Date:  2022-09-23 00:00:00
```

[5]:
```
# Listado categorías en la columna "category"
df["category"].unique()
```

[5]:
```
<StringArray>
[     'U.S. NEWS',         'COMEDY',      'PARENTING',     'WORLD NEWS',
 'CULTURE & ARTS',          'TECH',         'SPORTS',  'ENTERTAINMENT',
       'POLITICS',     'WEIRD NEWS',    'ENVIRONMENT',      'EDUCATION',
          'CRIME',        'SCIENCE',       'WELLNESS',       'BUSINESS',
 'STYLE & BEAUTY',   'FOOD & DRINK',          'MEDIA',   'QUEER VOICES',
  'HOME & LIVING',          'WOMEN',   'BLACK VOICES',         'TRAVEL',
          'MONEY',       'RELIGION',  'LATINO VOICES',         'IMPACT',
       'WEDDINGS',        'COLLEGE',        'PARENTS', 'ARTS & CULTURE',
          'STYLE',          'GREEN',         'TASTE', 'HEALTHY LIVING',
 'THE WORLDPOST',      'GOOD NEWS',      'WORLDPOST',          'FIFTY',
           'ARTS',        'DIVORCE']
Length: 42, dtype: str
```

1.2 2. Exploración inicial del corpus

El dataset completo contiene muchas categorías. Para una práctica guiada, es recomendable trabajar con un subconjunto reducido.

Esto permite que:

- el entrenamiento sea más rápido,
- los tópicos sean más fáciles de interpretar,
- y podamos comparar los tópicos aprendidos con categorías reales conocidas.

Aunque LDA es un método no supervisado, las categorías reales nos ayudan a comprobar si los tópicos descubiertos tienen sentido.

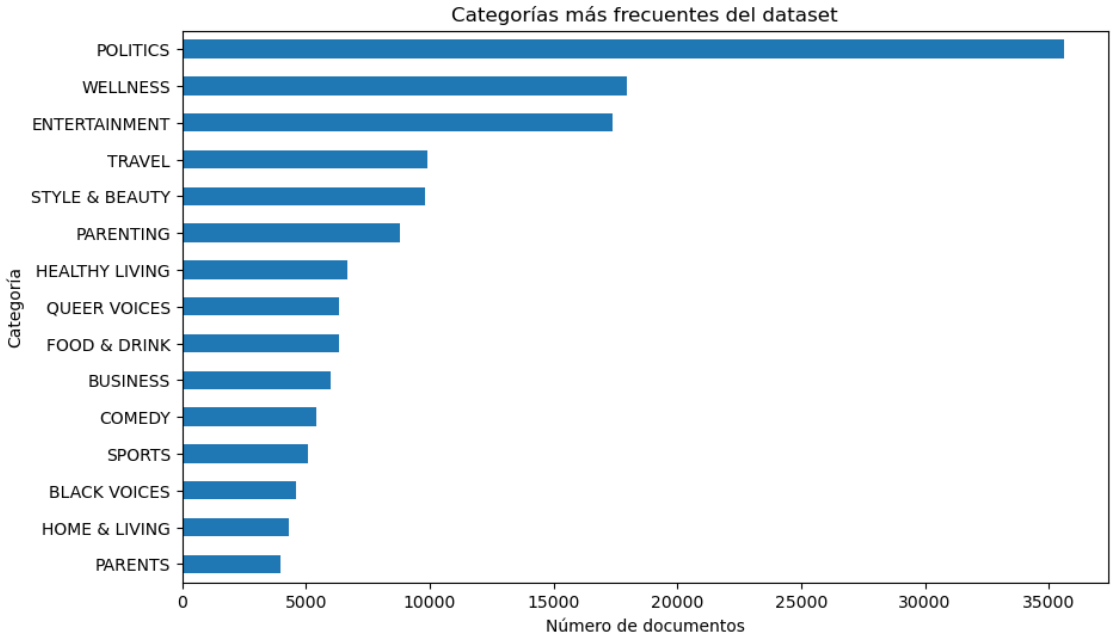
[6]:
```
# Número de documentos por categoría
conteo_categorias = df["category"].value_counts()

print("Número de categorías:", conteo_categorias.shape[0])
conteo_categorias.head(20)
```

```
Número de categorías: 42
```

[6]:
```
category
POLITICS          35602
WELLNESS          17945
ENTERTAINMENT     17362
TRAVEL             9900
STYLE & BEAUTY     9814
PARENTING          8791
HEALTHY LIVING     6694
QUEER VOICES       6347
FOOD & DRINK       6340
BUSINESS           5992
COMEDY             5400
SPORTS             5077
BLACK VOICES       4583
```

```
HOME & LIVING      4320
PARENTS            3955
THE WORLDPOST      3664
WEDDINGS           3653
WOMEN              3572
CRIME              3562
IMPACT             3484
Name: count, dtype: int64
```

[7]:
```python
# Visualización de las categorías más frecuentes
top_categorias = conteo_categorias.head(15)

plt.figure(figsize=(10, 6))
top_categorias.sort_values().plot(kind="barh")
plt.title("Categorías más frecuentes del dataset")
plt.xlabel("Número de documentos")
plt.ylabel("Categoría")
plt.show()
```



1.2.1 2.1. Selección de categorías para la práctica

Para esta práctica seleccionaremos algunas categorías con temas relativamente diferenciables.

La selección puede ajustarse si el dataset concreto tiene nombres ligeramente distintos de categorías.

La idea es trabajar con noticias de varios ámbitos para que LDA pueda descubrir estructuras temáticas.

[8]:
```
# Categorías propuestas para la sesión guiada.
categorias_propuestas = [
    "POLITICS",
    "SPORTS",
    "BUSINESS",
    "TRAVEL",
    "FOOD & DRINK",
    "WELLNESS",
    "ENTERTAINMENT",
    "TECH"
]

categorias_disponibles = []
for categoria in categorias_propuestas:
    if categoria in df["category"].unique():
        categorias_disponibles.append(categoria)

print("Categorías seleccionadas disponibles:")
print(categorias_disponibles)
```

```
Categorías seleccionadas disponibles:
['POLITICS', 'SPORTS', 'BUSINESS', 'TRAVEL', 'FOOD & DRINK', 'WELLNESS',
'ENTERTAINMENT', 'TECH']
```

[9]:
```
# Filtramos el dataset para quedarnos con las categorías seleccionadas
df_filtrado = df[df["category"].isin(categorias_disponibles)].copy()

print("Dimensiones tras filtrar categorías:", df_filtrado.shape)
df_filtrado["category"].value_counts()
```

```
Dimensiones tras filtrar categorías: (100322, 6)
```

[9]:
```
category
POLITICS         35602
WELLNESS         17945
ENTERTAINMENT    17362
TRAVEL            9900
FOOD & DRINK      6340
BUSINESS          5992
SPORTS            5077
TECH              2104
Name: count, dtype: int64
```

1.2.2 2.2. Reducción del tamaño del corpus

Para que el notebook se ejecute de forma rápida durante la sesión, tomaremos un máximo de documentos por categoría.

Esto no es obligatorio en un análisis real, pero resulta útil en clase.

[10]:
```
print(df_filtrado.columns)
```

```
Index(['link', 'headline', 'category', 'short_description', 'authors', 'date'],
dtype='str')
```

[11]:
```
# Número máximo de documentos por categoría.
# Podemos aumentar este valor si el entorno de ejecución lo permite.
"""
max_documentos_por_categoria = 500

df_reducido = (
    df_filtrado
    .groupby("category", group_keys=False)
    .apply(lambda datos_categoria: datos_categoria.sample(
        n=min(len(datos_categoria), max_documentos_por_categoria),
        random_state=42
    ))
    .reset_index(drop=True)
)

print("Dimensiones del corpus reducido:", df_reducido.shape)
df_reducido["category"].value_counts()
"""

# Número máximo de documentos por categoría.
# Podemos aumentar este valor si el entorno de ejecución lo permite.
max_documentos_por_categoria: int = 500

# Creamos una lista para almacenar los subconjuntos de cada categoría.
subconjuntos: list = []

# Recorremos cada categoría y sus documentos asociados.
for categoria, datos_categoria in df_filtrado.groupby("category"):

    # Calculamos cuántos documentos vamos a tomar de esta categoría.
    # Si hay menos de 500, tomamos todos.
    numero_documentos: int = min(
        len(datos_categoria),
        max_documentos_por_categoria
    )

    # Seleccionamos aleatoriamente documentos de esta categoría.
    datos_categoria_reducidos = datos_categoria.sample(
        n=numero_documentos,
        random_state=42
    )

    # Añadimos el subconjunto a la lista.
```

```
    subconjuntos.append(datos_categoria_reducidos)

# Unimos todos los subconjuntos en un único DataFrame.
df_reducido = pd.concat(subconjuntos, ignore_index=True)

print("Dimensiones del corpus reducido:", df_reducido.shape)
print(df_reducido["category"].value_counts())
```

```
Dimensiones del corpus reducido: (4000, 6)
category
BUSINESS         500
ENTERTAINMENT    500
FOOD & DRINK     500
POLITICS         500
SPORTS           500
TECH             500
TRAVEL           500
WELLNESS         500
Name: count, dtype: int64
```

1.3 3. Construcción del texto de trabajo

El dataset tiene varios campos textuales. Para esta práctica combinaremos:

- el titular (`headline`),
- y la descripción breve (`short_description`).

De esta forma, cada documento tendrá algo más de contenido que si usáramos únicamente el titular.

[12]:
```
# Sustituimos valores ausentes por cadenas vacías
df_reducido["headline"] = df_reducido["headline"].fillna("")
df_reducido["short_description"] = df_reducido["short_description"].fillna("")

# Creamos una columna de texto combinando titular y descripción
df_reducido["text"] = (
    df_reducido["headline"].astype(str)
    + ". "
    + df_reducido["short_description"].astype(str)
)

# Eliminamos espacios sobrantes
df_reducido["text"] = df_reducido["text"].str.strip()

df_reducido[["category", "headline", "short_description", "text"]].head()
```

[12]:
```
   category                                           headline  \
0  BUSINESS                  How to Manage Your Personal Brand
1  BUSINESS  It Looks Like Uber's Winning Its War With New …
2  BUSINESS      The Progressive Promise of Today's Technology
```

```
3  BUSINESS   Don't Let These 5 Confusing Words Mar Your Image
4  BUSINESS        What You Don't Know About Overnight Success

                                    short_description  \
0  Make no mistake: If you have a Facebook accoun…
1                                    Grab the popcorn.
2  A digital policy for the new century, tailored…
3  Tom's an articulate physician, totally able to…
4  I've been fighting this thing for 32 years. "O…

                                                text
0  How to Manage Your Personal Brand. Make no mis…
1  It Looks Like Uber's Winning Its War With New …
2  The Progressive Promise of Today's Technology…
3  Don't Let These 5 Confusing Words Mar Your Ima…
4  What You Don't Know About Overnight Success. I…
```

[13]:
```
print("Ejemplo de noticia:")
print("- Headline: ", df_reducido.iloc[0]["headline"])
print("- Short description: ", df_reducido.iloc[0]["short_description"])
print("- Text: ", df_reducido.iloc[0]["text"])
```

```
Ejemplo de noticia:
- Headline:  How to Manage Your Personal Brand
- Short description:  Make no mistake: If you have a Facebook account, an
Instagram page, a Twitter profile, you are a brand. Every time you upload a
photo, add a link, or post an update, you're putting into the world another idea
of yourself and what you stand for.
- Text:  How to Manage Your Personal Brand. Make no mistake: If you have a
Facebook account, an Instagram page, a Twitter profile, you are a brand. Every
time you upload a photo, add a link, or post an update, you're putting into the
world another idea of yourself and what you stand for.
```

1.3.1 3.1. Revisión de textos vacíos o demasiado cortos

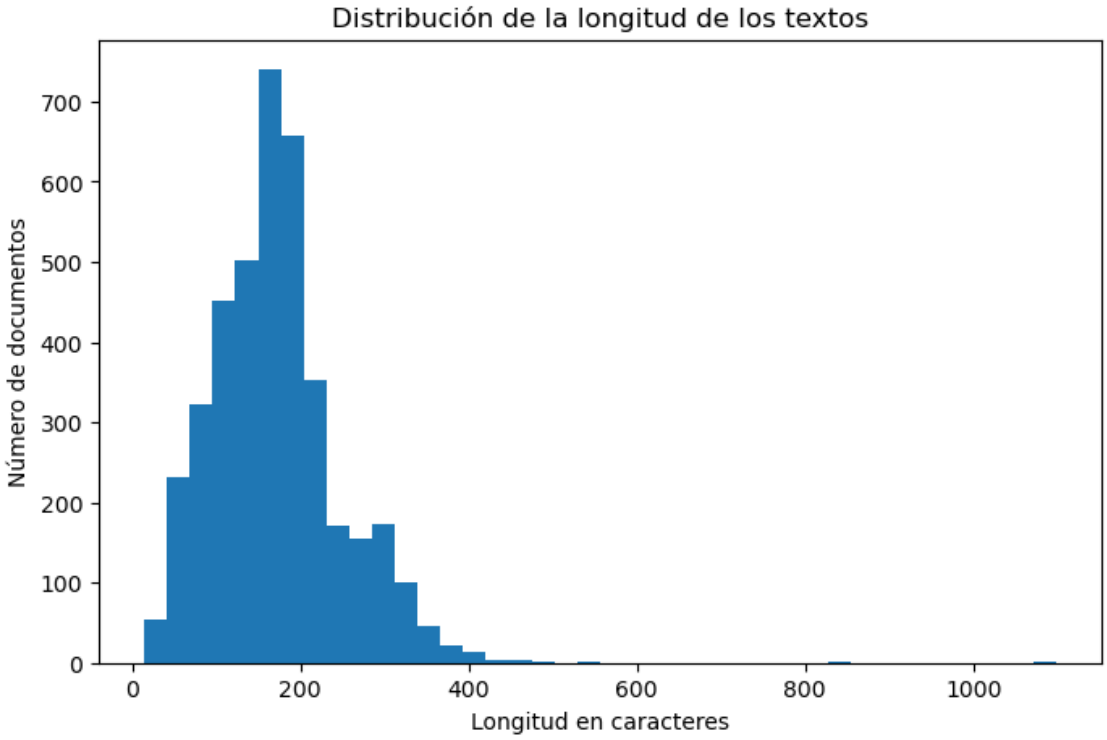
Los documentos muy cortos pueden ser problemáticos para LDA porque contienen poca información para inferir tópicos.

Por ello, revisaremos la longitud de los textos y eliminaremos aquellos demasiado breves.

[14]:
```
# Calculamos una longitud simple en número de caracteres
df_reducido["text_length"] = df_reducido["text"].str.len()

print(df_reducido["text_length"].describe())

plt.figure(figsize=(8, 5))
plt.hist(df_reducido["text_length"], bins=40)
plt.title("Distribución de la longitud de los textos")
```

```
plt.xlabel("Longitud en caracteres")
plt.ylabel("Número de documentos")
plt.show()
```

```
count    4000.00000
mean      171.04225
std        75.56621
min        14.00000
25%       119.00000
50%       166.00000
75%       206.00000
max      1098.00000
Name: text_length, dtype: float64
```



[15]:
```
# Eliminamos textos demasiado cortos
longitud_minima = 40

df_reducido = df_reducido[df_reducido["text_length"] >= longitud_minima].copy()
df_reducido = df_reducido.reset_index(drop=True)

print("Dimensiones tras eliminar textos cortos:", df_reducido.shape)
```

```
Dimensiones tras eliminar textos cortos: (3952, 8)
```

1.4 4. Preprocesamiento básico

El objetivo del preprocesamiento es reducir ruido antes de construir la representación documento-término.

En esta práctica aplicaremos un preprocesamiento sencillo:

- Conversión a minúsculas,
- Eliminación de caracteres no alfabéticos,
- Tokenización básica,
- Eliminación de stopwords en inglés,
- Eliminación de tokens muy cortos.

No aplicaremos stemming como flujo principal, porque puede producir raíces poco interpretables.

La lematización se deja como extensión opcional al final del notebook.

[16]:
```
import nltk

# Descargamos la lista de stopwords de NLTK.
# Si ya está descargada, NLTK no la vuelve a descargar.
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/sergio/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[16]:
```
True
```

[17]:
```
from nltk.corpus import stopwords

stopwords_ingles = set(stopwords.words("english"))

# Añadimos algunas palabras frecuentes poco informativas en noticias.
# Esta lista puede ajustarse tras inspeccionar resultados.
stopwords_adicionales = {
    "said", "says", "say", "also", "one", "two", "new", "would", "could",
    "get", "like", "time", "people", "year", "years", "make", "made"
}

stopwords_totales = stopwords_ingles.union(stopwords_adicionales)

print("Número total de stopwords:", len(stopwords_totales))
```

```
Número total de stopwords: 215
```

[18]:
```
stopwords_totales
```

[18]:
```
{'a',
 'about',
 'above',
```

```
'after',
'again',
'against',
'ain',
'all',
'also',
'am',
'an',
'and',
'any',
'are',
'aren',
"aren't",
'as',
'at',
'be',
'because',
'been',
'before',
'being',
'below',
'between',
'both',
'but',
'by',
'can',
'could',
'couldn',
"couldn't",
'd',
'did',
'didn',
"didn't",
'do',
'does',
'doesn',
"doesn't",
'doing',
'don',
"don't",
'down',
'during',
'each',
'few',
'for',
'from',
'further',
```

```
'get',
'had',
'hadn',
"hadn't",
'has',
'hasn',
"hasn't",
'have',
'haven',
"haven't",
'having',
'he',
"he'd",
"he'll",
"he's",
'her',
'here',
'hers',
'herself',
'him',
'himself',
'his',
'how',
'i',
"i'd",
"i'll",
"i'm",
"i've",
'if',
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it'd",
"it'll",
"it's",
'its',
'itself',
'just',
'like',
'll',
'm',
'ma',
'made',
'make',
```

```
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'new',
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'one',
'only',
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
'people',
're',
's',
'said',
'same',
'say',
'says',
'shan',
"shan't",
'she',
"she'd",
"she'll",
"she's",
'should',
"should've",
'shouldn',
```

```
"shouldn't",
'so',
'some',
'such',
't',
'than',
'that',
"that'll",
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'these',
'they',
"they'd",
"they'll",
"they're",
"they've",
'this',
'those',
'through',
'time',
'to',
'too',
'two',
'under',
'until',
'up',
've',
'very',
'was',
'wasn',
"wasn't",
'we',
"we'd",
"we'll",
"we're",
"we've",
'were',
'weren',
"weren't",
'what',
'when',
'where',
```

```
'which',
'while',
'who',
'whom',
'why',
'will',
'with',
'won',
"won't",
'would',
'wouldn',
"wouldn't",
'y',
'year',
'years',
'you',
"you'd",
"you'll",
"you're",
"you've",
'your',
'yours',
'yourself',
'yourselves'}
```

[19]:
```
def preprocesar_texto(texto):
    '''
    Aplica un preprocesamiento básico sobre un texto en inglés.

    Pasos:
    1. Convertir a minúsculas.
    2. Eliminar caracteres no alfabéticos.
    3. Separar en tokens.
    4. Eliminar stopwords.
    5. Eliminar tokens muy cortos.

    Devuelve:
    - Una cadena con los tokens procesados unidos por espacios.
    '''

    # 1. Convertir a minúsculas
    texto = texto.lower()
    # 2. Eliminar caracteres no alfabéticos (también descarta números). \s␣
 ↪significa un "espacio en blanco"
    texto = re.sub(r"[^a-z\s]", " ", texto)
    # 3. Separar en tokens, obviando espacios duplicados
    tokens = texto.split()
```

```
    # Filtramos aquellos tokens que no sean stopwords y cuya longitud (nº de letras) sea mayor o igual a 3
    tokens_filtrados = []
    for token in tokens:
        if token not in stopwords_totales and len(token) >= 3:
            tokens_filtrados.append(token)

    texto_procesado = " ".join(tokens_filtrados)

    return texto_procesado
```

[20]:
```
df_reducido["processed_text"] = df_reducido["text"].apply(preprocesar_texto)

df_reducido[["category", "text", "processed_text"]].head()
```

[20]:
```
    category                                               text  \
0  BUSINESS  How to Manage Your Personal Brand. Make no mis…
1  BUSINESS  It Looks Like Uber's Winning Its War With New …
2  BUSINESS  The Progressive Promise of Today's Technology…
3  BUSINESS  Don't Let These 5 Confusing Words Mar Your Ima…
4  BUSINESS  What You Don't Know About Overnight Success. I…

                                      processed_text
0  manage personal brand mistake facebook account…
1            looks uber winning war york grab popcorn
2  progressive promise today technology digital p…
3  let confusing words mar image tom articulate p…
4  know overnight success fighting thing overnigh…
```

1.4.1 4.1. Comparación entre texto original y texto procesado

Es importante revisar manualmente algunos ejemplos.

El preprocesamiento puede eliminar ruido, pero también puede eliminar información útil si se aplica de forma demasiado agresiva.

[21]:
```
for indice in range(3):
    print("=" * 100)
    print("Categoría:", df_reducido.loc[indice, "category"])
    print("\nTexto original:")
    print(df_reducido.loc[indice, "text"])
    print("\nTexto procesado:")
    print(df_reducido.loc[indice, "processed_text"])
    print()
```

```
====================================================================================================
Categoría: BUSINESS
```

```
Texto original:
How to Manage Your Personal Brand. Make no mistake: If you have a Facebook
account, an Instagram page, a Twitter profile, you are a brand. Every time you
upload a photo, add a link, or post an update, you're putting into the world
another idea of yourself and what you stand for.

Texto procesado:
manage personal brand mistake facebook account instagram page twitter profile
brand every upload photo add link post update putting world another idea stand

================================================================================
====================
Categoría: BUSINESS

Texto original:
It Looks Like Uber's Winning Its War With New York. Grab the popcorn.

Texto procesado:
looks uber winning war york grab popcorn

================================================================================
====================
Categoría: BUSINESS

Texto original:
The Progressive Promise of Today's Technology. A digital policy for the new
century, tailored not just to the moment but for the future, is vital if we are
to unleash economic growth, shared prosperity, and the full potential of
technology for citizens and consumers. But such a policy architecture requires a
new consensus--on privacy, on security, on customer protections, on growth and
mobility.

Texto procesado:
progressive promise today technology digital policy century tailored moment
future vital unleash economic growth shared prosperity full potential technology
citizens consumers policy architecture requires consensus privacy security
customer protections growth mobility
```

1.5 5. Exploración del texto procesado

Antes de entrenar LDA, revisaremos qué palabras aparecen con más frecuencia en el corpus procesado.

Esto ayuda a detectar:

- términos demasiado frecuentes,
- palabras poco informativas,

- posibles stopwords adicionales,
- y ruido que podría afectar a los tópicos.

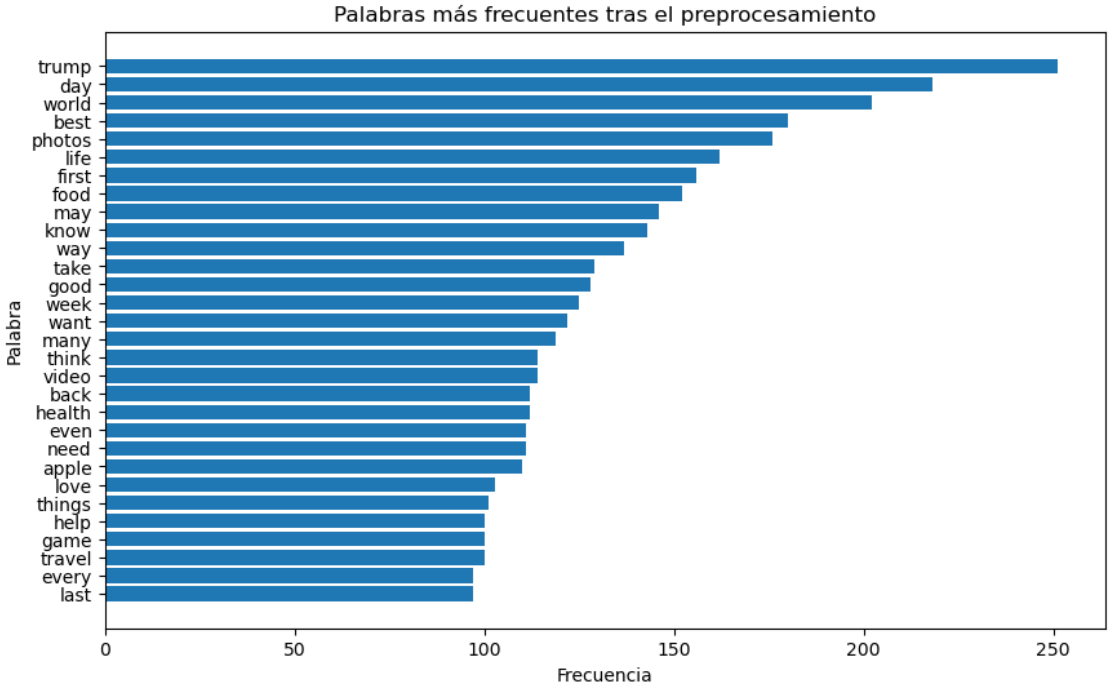
[22]:
```
# Unimos todos los textos procesados
todos_los_tokens = []

for texto in df_reducido["processed_text"]:
    tokens = texto.split()
    for token in tokens:
        todos_los_tokens.append(token)

# Calculamos el número de apariciones de cada token
frecuencias = Counter(todos_los_tokens)

# Obtenemos y mostramos las frecuencias de las 30 palabras más frecuentes
palabras_frecuentes = pd.DataFrame(
    frecuencias.most_common(30),
    columns=["word", "frequency"]
)

palabras_frecuentes
```

[22]:

	word	frequency
0	trump	251
1	day	218
2	world	202
3	best	180
4	photos	176
5	life	162
6	first	156
7	food	152
8	may	146
9	know	143
10	way	137
11	take	129
12	good	128
13	week	125
14	want	122
15	many	119
16	think	114
17	video	114
18	back	112
19	health	112
20	even	111
21	need	111
22	apple	110
23	love	103

24	things	101
25	help	100
26	game	100
27	travel	100
28	every	97
29	last	97

[23]:
```
plt.figure(figsize=(10, 6))
plt.barh(
    palabras_frecuentes["word"][::-1],
    palabras_frecuentes["frequency"][::-1]
)
plt.title("Palabras más frecuentes tras el preprocesamiento")
plt.xlabel("Frecuencia")
plt.ylabel("Palabra")
plt.show()
```



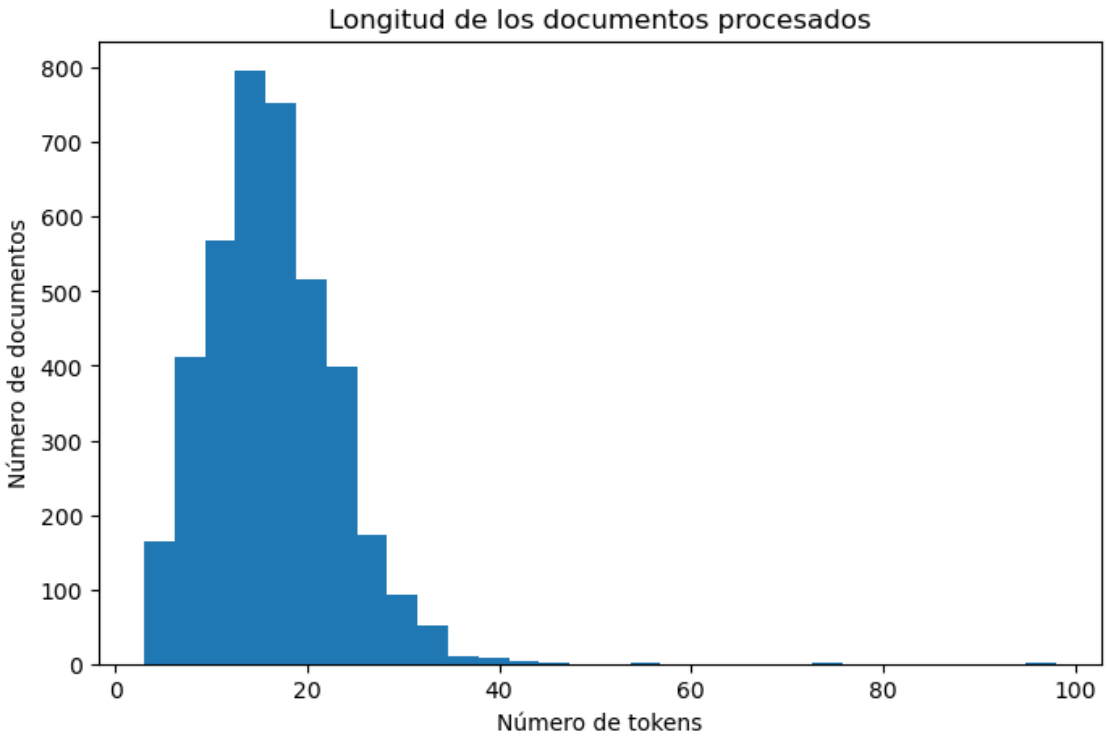
1.5.1 5.1. Longitud de los documentos procesados

También revisaremos cuántos tokens quedan en cada documento tras el preprocesamiento.

Si muchos documentos quedan con muy pocas palabras, el modelo de tópicos tendrá menos información para trabajar.

[24]:
```
df_reducido["processed_length"] = df_reducido["processed_text"].apply(lambda
 ↪texto: len(texto.split()))

print(df_reducido["processed_length"].describe())

plt.figure(figsize=(8, 5))
plt.hist(df_reducido["processed_length"], bins=30)
plt.title("Longitud de los documentos procesados")
plt.xlabel("Número de tokens")
plt.ylabel("Número de documentos")
plt.show()
```

```
count    3952.000000
mean       16.234818
std         6.553792
min         3.000000
25%        12.000000
50%        16.000000
75%        20.000000
max        98.000000
Name: processed_length, dtype: float64
```



[25]:
```
# Eliminamos documentos que hayan quedado con muy pocos tokens tras el␣
 ↪preprocesamiento
min_tokens = 5

df_reducido = df_reducido[df_reducido["processed_length"] >= min_tokens].copy()
df_reducido = df_reducido.reset_index(drop=True)

print("Dimensiones tras filtrar por longitud procesada:", df_reducido.shape)
```

```
Dimensiones tras filtrar por longitud procesada: (3924, 10)
```

1.6 6. Representación de la matriz documento-término con conteos

Para entrenar LDA clásico usaremos una matriz documento-término con conteos.

Esto significa que:

- Cada fila representa un documento
- Cada columna representa un término del vocabulario
- Cada celda indica cuántas veces aparece un término en un documento

Usamos conteos porque LDA intenta explicar las ocurrencias observadas de palabras en los documentos.

Por eso, en el flujo principal no usaremos TF-IDF como entrada de LDA.

Para construir esta matriz usaremos `CountVectorizer`, que se encarga de crear el vocabulario y contar las apariciones de cada término en cada documento.

En esta práctica configuraremos algunos parámetros importantes:

- `lowercase=False`: indica que `CountVectorizer` no debe convertir el texto a minúsculas, porque ya lo hemos hecho previamente durante el preprocesamiento.
- `min_df=5`: conserva únicamente los términos que aparecen en al menos 5 documentos distintos. Esto ayuda a eliminar palabras muy raras, errores o términos demasiado específicos.
- `max_df=0.80`: elimina los términos que aparecen en más del 80% de los documentos. Esto ayuda a quitar palabras demasiado frecuentes que no aportan mucha información para distinguir tópicos.
- `max_features=3000`: limita el vocabulario a un máximo de 3000 términos, seleccionando los más frecuentes después de aplicar los filtros anteriores. Esto reduce el tamaño de la matriz y hace que el entrenamiento sea más manejable.

Es importante no confundir `min_df=5` con el filtro anterior `min_tokens=5`.

Aunque en ambos casos aparece el valor 5, se aplican a niveles diferentes:

- `min_tokens=5` filtra documentos: elimina documentos que han quedado con menos de 5 palabras útiles tras el preprocesamiento.
- `min_df=5` filtra términos: elimina palabras que aparecen en menos de 5 documentos distintos.

Por tanto, `min_tokens` controla que cada documento tenga una cantidad mínima de información, mientras que `min_df` controla que el vocabulario no incluya términos demasiado raros.

Ambos filtros ayudan a reducir ruido, pero actúan sobre elementos distintos del corpus.

[26]:
```
from sklearn.feature_extraction.text import CountVectorizer

# CountVectorizer construye automáticamente el vocabulario y la matriz␣
 ↪documento-término.
# Como ya hemos preprocesado el texto, indicamos lowercase=False.
vectorizador_conteos = CountVectorizer(
    lowercase=False,
    min_df=5,
    max_df=0.80,
    max_features=3000
)

matriz_conteos = vectorizador_conteos.
 ↪fit_transform(df_reducido["processed_text"])

print("Dimensiones de la matriz documento-término (nº documentos, nº términos):
 ↪")
print(matriz_conteos.shape)
```

```
Dimensiones de la matriz documento-término (nº documentos, nº términos):
(3924, 2797)
```

[27]:
```
vocabulario = vectorizador_conteos.get_feature_names_out()

print("Tamaño del vocabulario:", len(vocabulario))
print("Primeros términos del vocabulario:")
print(vocabulario[:50])
```

```
Tamaño del vocabulario: 2797
Primeros términos del vocabulario:
['abc' 'ability' 'able' 'abortion' 'absolutely' 'abuse' 'academy'
 'accepted' 'access' 'accessible' 'accident' 'according' 'account'
 'accusations' 'accused' 'across' 'act' 'acting' 'action' 'actions'
 'active' 'activities' 'activity' 'actor' 'actors' 'actress' 'actual'
 'actually' 'add' 'added' 'adding' 'addition' 'address' 'adds'
 'administration' 'admits' 'admitted' 'adorable' 'ads' 'adult' 'adults'
 'advance' 'advanced' 'advances' 'advantage' 'adventure' 'advertising'
 'advice' 'adviser' 'affect']
```

1.6.1 6.1. Ejemplo de vector de un documento

La matriz generada es dispersa, porque la mayoría de documentos solo contienen una pequeña parte del vocabulario.

Vamos a inspeccionar el vector de un documento concreto mostrando únicamente los términos con frecuencia mayor que cero.

[28]:
```
# Podemos modificar este valor para indicar el índice del documento concreto a␣
 ↪visualizar
```

```
indice_documento = 0

# Extrae la fila de la matriz asociada al documento "indice_documento"
vector_documento = matriz_conteos[indice_documento].toarray()[0]

# Lista de términos del documento
terminos_presentes = []

# Recorre el vector del documento posición a posición, y añade el término a␣
 ↪"terminos_presentes" si la frecuencia es > 0
# como una tupla junto con su frecuencia
# Ejemplo: ("market", 2)
for indice_termino, frecuencia in enumerate(vector_documento):
    if frecuencia > 0:
        terminos_presentes.append((vocabulario[indice_termino], frecuencia))

# Convierte la lista de términos presentes en un dataframe de pandas, con␣
 ↪columnas "term" y "count", de forma ordenada por frecuencia
df_vector_documento = pd.DataFrame(
    terminos_presentes,
    columns=["term", "count"]
).sort_values(by="count", ascending=False)

print("Categoría real:", df_reducido.loc[indice_documento, "category"])
print("Texto original:")
print(df_reducido.loc[indice_documento, "text"])
print("\nTérminos presentes en la representación:")
df_vector_documento.head(20)
```

```
Categoría real: BUSINESS
Texto original:
How to Manage Your Personal Brand. Make no mistake: If you have a Facebook
account, an Instagram page, a Twitter profile, you are a brand. Every time you
upload a photo, add a link, or post an update, you're putting into the world
another idea of yourself and what you stand for.

Términos presentes en la representación:
```

[28]:

	term	count
3	brand	2
0	account	1
11	page	1
19	update	1
18	twitter	1
17	stand	1
16	putting	1
15	profile	1

```
14       post      1
13      photo      1
12   personal      1
10    mistake      1
1         add      1
9      manage      1
8        link      1
7   instagram      1
6        idea      1
5    facebook      1
4       every      1
2     another      1
```

1.7 7. Entrenamiento de un modelo LDA

A continuación entrenaremos un modelo LDA.

Para comenzar, fijaremos manualmente el número de tópicos. Este valor es un hiperparámetro del modelo.

En una práctica real, conviene probar varios valores y comparar los resultados.

Para crear el modelo usaremos `LatentDirichletAllocation`, indicando algunos parámetros importantes:

- `n_components=numero_topicos`: indica cuántos tópicos queremos que aprenda el modelo. Si `numero_topicos = 8`, LDA intentará descubrir 8 tópicos distintos en el corpus.
- `random_state=42`: fija la semilla aleatoria del modelo para que los resultados sean reproducibles. Así, si ejecutamos el notebook varias veces con los mismos datos y parámetros, obtendremos resultados más estables.
- `learning_method="batch"`: indica que el modelo usará todos los documentos en cada iteración de entrenamiento. Es una opción adecuada para esta práctica porque resulta más estable y sencilla de explicar.
- `max_iter=20`: indica el número máximo de iteraciones de entrenamiento. En cada iteración, el modelo reajusta internamente las distribuciones de tópicos por documento y de palabras por tópico.

Estos parámetros permiten controlar el comportamiento del entrenamiento.

El más importante desde el punto de vista interpretativo es `n_components`, porque determina cuántos tópicos intentará descubrir el modelo.

[29]:
```python
from sklearn.decomposition import LatentDirichletAllocation

numero_topicos = 8

modelo_lda = LatentDirichletAllocation(
    n_components=numero_topicos,
    random_state=42,
    learning_method="batch",
```

```
    max_iter=20
)

modelo_lda.fit(matriz_conteos)

print("Modelo LDA entrenado")
```

```
Modelo LDA entrenado
```

1.7.1 7.1. Palabras principales de cada tópico

Cada tópico aprendido por LDA puede representarse mediante las palabras con mayor peso dentro de ese tópico.

Estas palabras no son una etiqueta automática. Sirven para que una persona interprete el tópico.

[30]:
```
def obtener_palabras_topico(modelo, vocabulario, numero_palabras=10):
    '''
    Devuelve una tabla con las palabras principales de cada tópico.
    '''
    filas = []

    for indice_topico, distribucion_palabras in enumerate(modelo.components_):
        indices_ordenados = distribucion_palabras.argsort()[::-1]
        indices_top = indices_ordenados[:numero_palabras]

        palabras = []
        pesos = []

        for indice in indices_top:
            palabras.append(vocabulario[indice])
            pesos.append(distribucion_palabras[indice])

        filas.append({
            "topic": indice_topico,
            "top_words": ", ".join(palabras),
            "weights": pesos
        })

    return pd.DataFrame(filas)

tabla_topicos = obtener_palabras_topico(modelo_lda, vocabulario,␣
 ↪numero_palabras=12)
tabla_topicos[["topic", "top_words"]]
```

[30]:

	topic	top_words
0	0	trump, donald, president, obama, clinton, hous…
1	1	first, star, life, way, uber, data, nfl, olymp…

```
2      2  city, things, photos, york, best, study, every…
3      3  apple, week, iphone, back, help, best, travel,…
4      4  food, photos, facebook, think, holiday, well, …
5      5  health, healthy, may, keep, take, want, good, …
6      6  world, day, first, state, little, best, play, …
7      7  food, super, day, watch, summer, week, bowl, v…
```

[31]:
```
categorias_propuestas = [
    "POLITICS",
    "SPORTS",
    "BUSINESS",
    "TRAVEL",
    "FOOD & DRINK",
    "WELLNESS",
    "ENTERTAINMENT",
    "TECH"
]
```

1.8 8. Interpretación de tópicos

La interpretación de tópicos combina la salida del modelo con revisión humana.

Para interpretar un tópico conviene revisar:

- sus palabras principales,
- los documentos donde ese tópico tiene más peso,
- y si el conjunto de palabras tiene sentido para el objetivo del análisis.

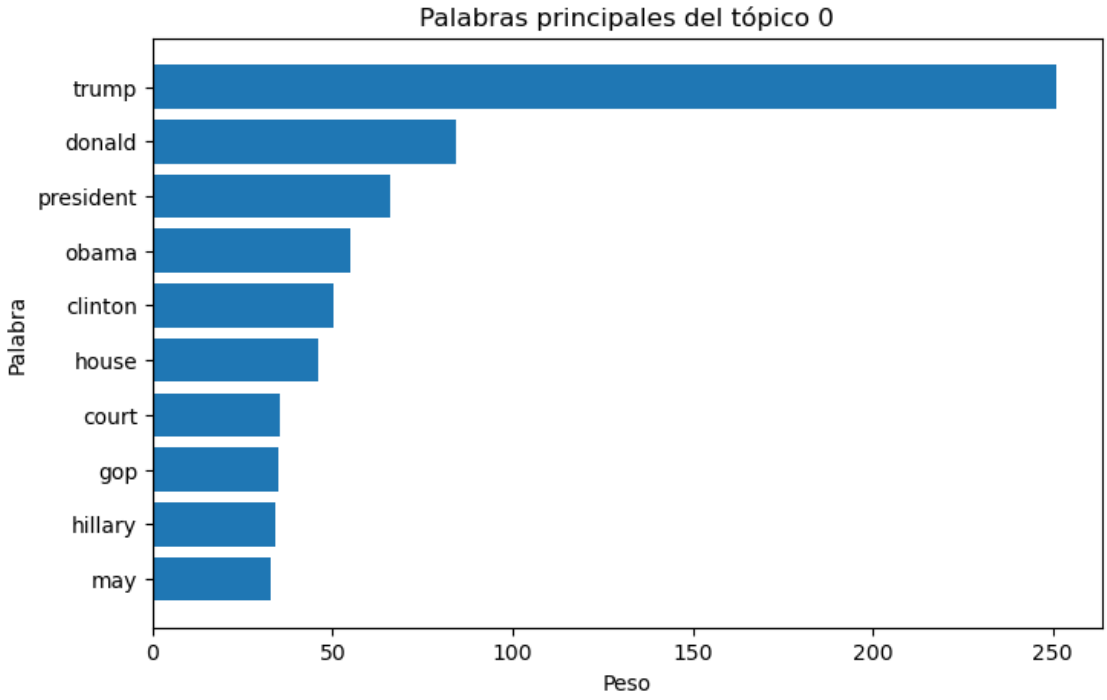
1.8.1 8.1. Visualización de palabras principales por tópico

Una forma sencilla de analizar un tópico es representar sus palabras principales con un gráfico de barras.

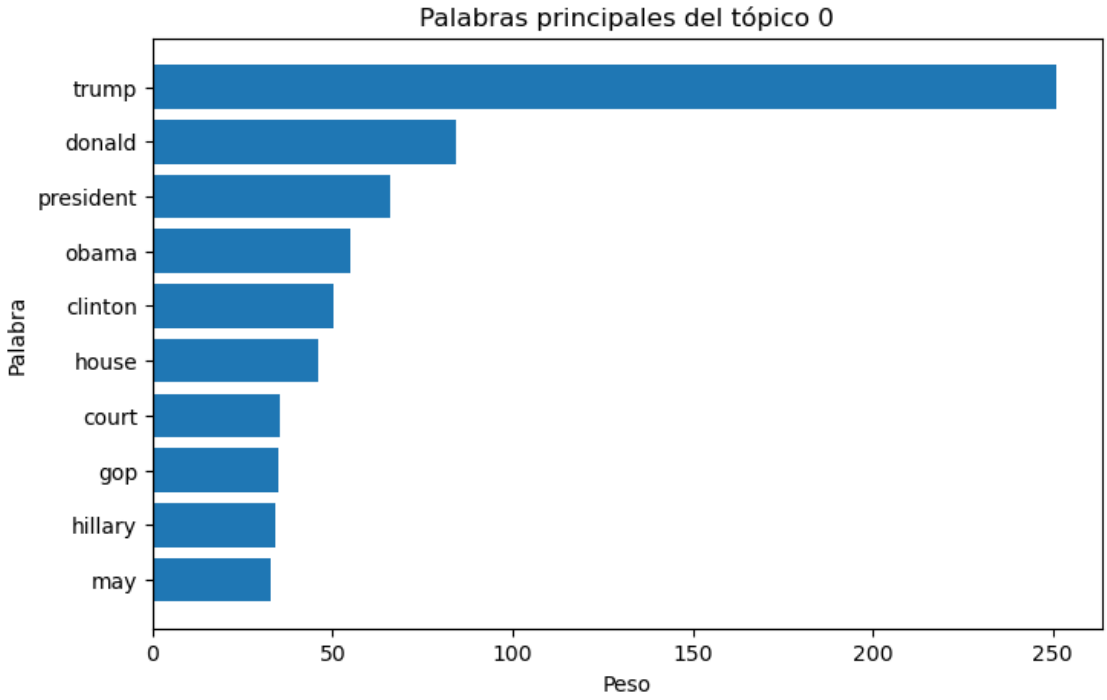
Esto ayuda a ver qué términos dominan cada tópico.

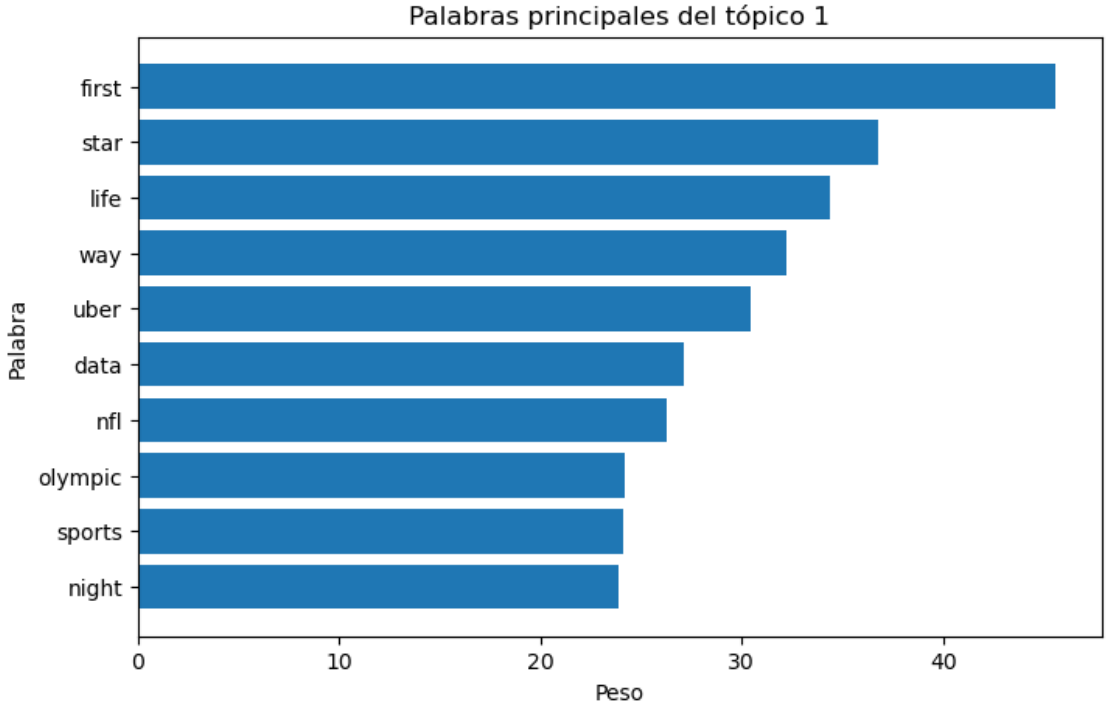
[32]:
```
def graficar_topico(modelo, vocabulario, indice_topico, numero_palabras=10):
    distribucion_palabras = modelo.components_[indice_topico]
    indices_ordenados = distribucion_palabras.argsort()[::-1]
    indices_top = indices_ordenados[:numero_palabras]

    palabras = vocabulario[indices_top]
    pesos = distribucion_palabras[indices_top]

    plt.figure(figsize=(8, 5))
    plt.barh(palabras[::-1], pesos[::-1])
    plt.title(f"Palabras principales del tópico {indice_topico}")
    plt.xlabel("Peso")
    plt.ylabel("Palabra")
    plt.show()
```

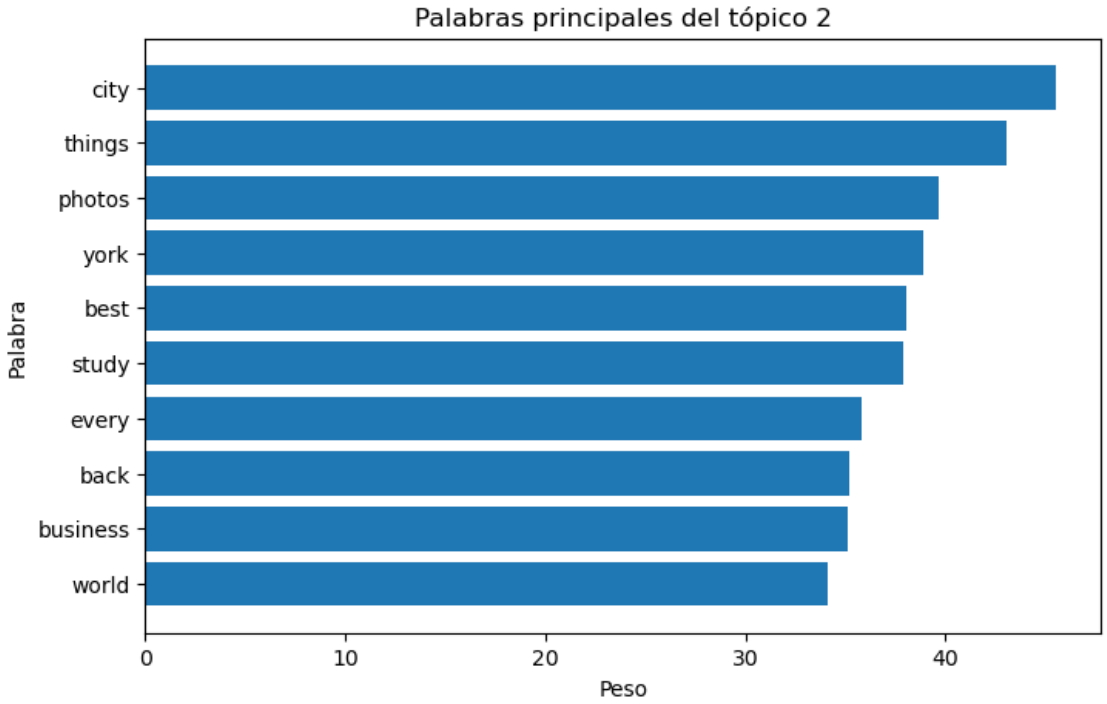
```
# Ejemplo: mostramos el tópico 0
graficar_topico(modelo_lda, vocabulario, indice_topico=0, numero_palabras=10)
```

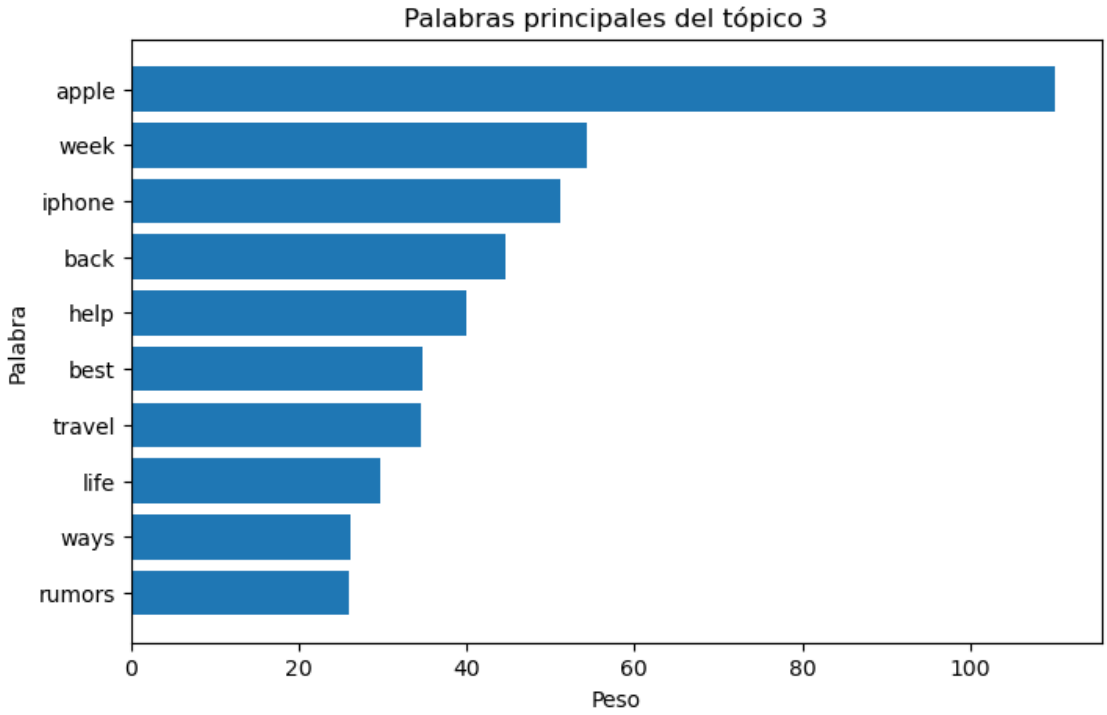


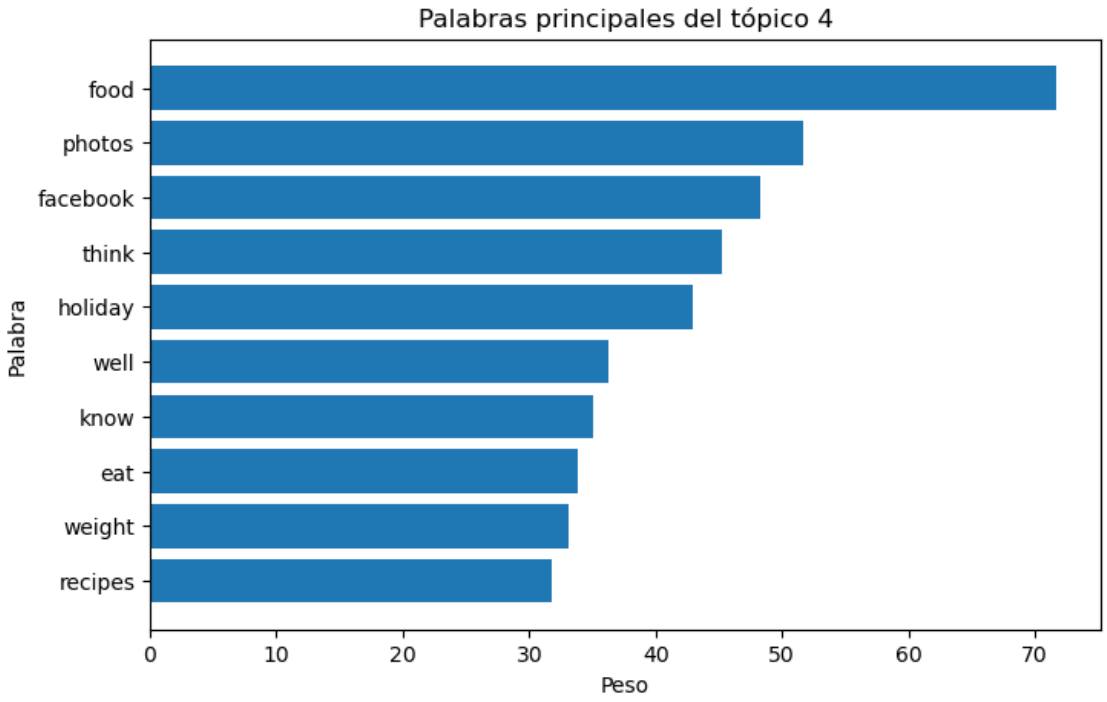
[33]:
```
# Mostramos todos los tópicos uno a uno
for indice_topico in range(numero_topicos):
    graficar_topico(modelo_lda, vocabulario, indice_topico=indice_topico,␣
 ↪numero_palabras=10)
```

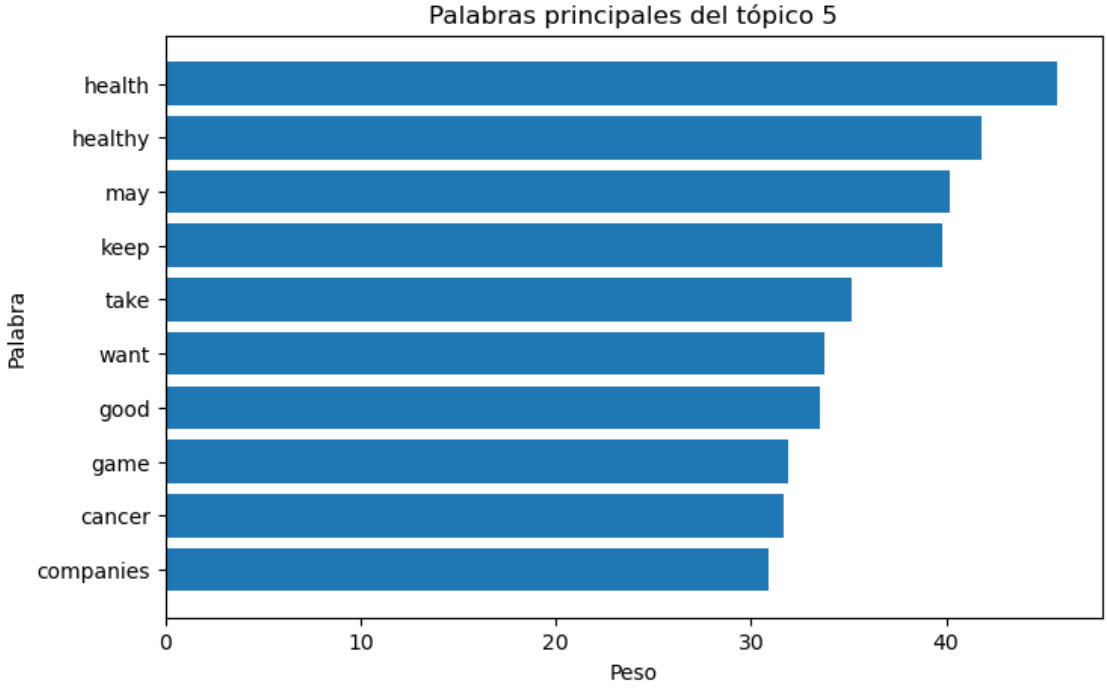

Palabras principales del tópico 0
trump
donald
president
obama
clinton
house
court
gop
hillary
may
Palabra
0
50
100
150
200
250
Peso

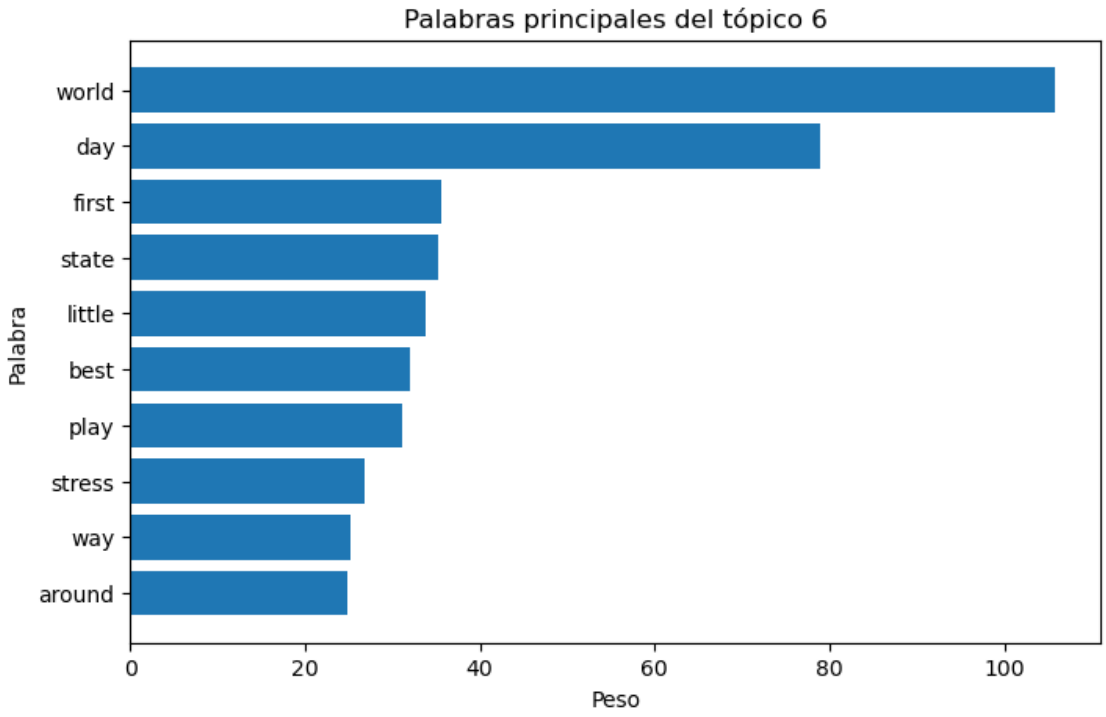

Palabras principales del tópico 1
first
star
life
way
uber
data
nfl
olympic
sports
night
Palabra
0
10
20
30
40
Peso

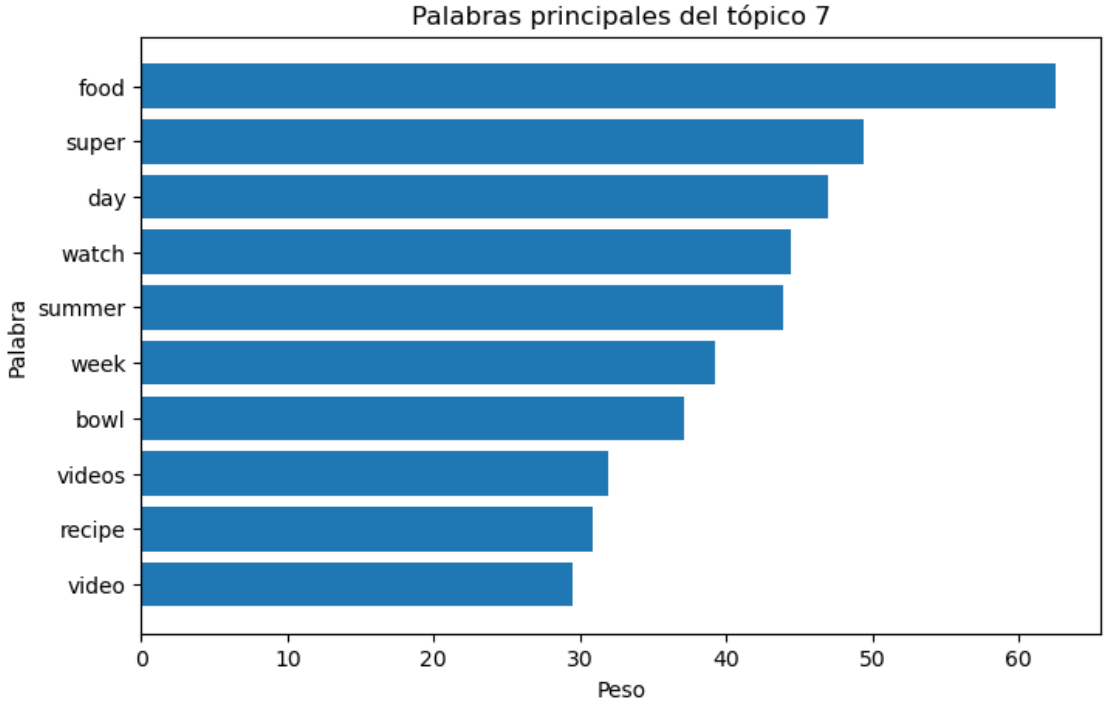

Palabras principales del tópico 2
city
things
photos
york
best
study
every
back
business
world
Palabra
0
10
20
30
40
Peso


Palabras principales del tópico 3
apple
week
iphone
back
help
best
travel
life
ways
rumors
Palabra
0
20
40
60
80
100
Peso


Palabras principales del tópico 4
food
photos
facebook
think
holiday
well
know
eat
weight
recipes
Palabra
0
10
20
30
40
50
60
70
Peso


Palabras principales del tópico 5
health
healthy
may
keep
take
want
good
game
cancer
companies
Palabra
0
10
20
30
40
Peso


Palabras principales del tópico 6
world
day
first
state
little
best
play
stress
way
around
Palabra
0
20
40
60
80
100
Peso


Palabras principales del tópico 7
food
super
day
watch
summer
week
bowl
videos
recipe
video
Palabra
0
10
20
30
40
50
60
Peso

1.9 9. Distribución de tópicos por documento

LDA no asigna necesariamente un único tópico a cada documento.

En su lugar, estima una distribución de tópicos para cada documento.

Esto significa que cada documento se representa como una combinación de los tópicos aprendidos por el modelo.

Por ejemplo:

- Documento A: 70% política, 20% economía, 10% viajes.
- Documento B: 60% deporte, 25% entretenimiento, 15% política.

Para obtener esta información usaremos el método `transform()` del modelo LDA:

`distribucion_documentos = modelo_lda.transform(matriz_conteos)`

Este método aplica el modelo LDA ya entrenado sobre la matriz documento-término con conteos.

El resultado es una matriz documento-tópico.

En esta matriz:

- Cada fila representa un documento.
- Cada columna representa un tópico aprendido por LDA.
- Cada celda contiene la proporción o peso de ese tópico en ese documento.

Por ejemplo, si el corpus tiene 3924 documentos y el modelo se ha entrenado con 8 tópicos, la matriz tendrá dimensiones `(3924, 8)`.

Esto significa que cada documento queda representado mediante 8 valores, uno por cada tópico.

Cada fila suma aproximadamente 1, porque representa una distribución de probabilidad sobre los tópicos.

Por ejemplo, una fila podría ser:

`[0.05, 0.02, 0.70, 0.04, 0.03, 0.10, 0.03, 0.03]`

En ese caso, el tópico dominante del documento sería el tópico 2 (tercera columna), porque es el que tiene mayor peso.

A continuación obtendremos esta distribución para todos los documentos del corpus.

[34]:
```
# Transformamos la matriz de conteos para obtener la distribución de tópicos␣
 ↪por documento
distribucion_documentos = modelo_lda.transform(matriz_conteos)

# Salida: matriz con forma (nº documentos, nº tópicos).
# Cada fila representa un documento.
# Cada columna representa un tópico.
# Cada celda contiene la proporción o peso de ese tópico en ese documento.
# Los valores de cada fila suman aproximadamente 1.
print("Dimensiones de la matriz documento-tópico:")
print(distribucion_documentos.shape)
```

```
Dimensiones de la matriz documento-tópico:
(3924, 8)
```

[35]:
```
# Creamos columnas con el peso de cada tópico en cada documento

# Variables de partida:
# - df_reducido: información original y procesada de cada noticia
# - distribucion_documentos: distribución de tópicos que LDA ha estimado por␣
 ↪documento

df_topicos_documentos = df_reducido[["category", "headline",␣
 ↪"short_description", "text", "processed_text"]].copy()

# Para cada tópico crea una nueva columna topic_nº y la añade a los datos␣
 ↪procesados de partida
for indice_topico in range(numero_topicos):
    nombre_columna = f"topic_{indice_topico}"
    df_topicos_documentos[nombre_columna] = distribucion_documentos[:,␣
 ↪indice_topico]

# Añadimos dos columnas relacionadas con el tópico mayoritario (nº de tópico y␣
 ↪su peso)
df_topicos_documentos["dominant_topic"] = distribucion_documentos.argmax(axis=1)
df_topicos_documentos["dominant_topic_weight"] = distribucion_documentos.
 ↪max(axis=1)

df_topicos_documentos.head()
```

[35]:

	category	headline	\
0	BUSINESS	How to Manage Your Personal Brand	
1	BUSINESS	It Looks Like Uber's Winning Its War With New …	
2	BUSINESS	The Progressive Promise of Today's Technology	
3	BUSINESS	Don't Let These 5 Confusing Words Mar Your Image	
4	BUSINESS	What You Don't Know About Overnight Success	

	short_description	\
0	Make no mistake: If you have a Facebook accoun…	
1	Grab the popcorn.	
2	A digital policy for the new century, tailored…	
3	Tom's an articulate physician, totally able to…	
4	I've been fighting this thing for 32 years. "O…	

	text	\
0	How to Manage Your Personal Brand. Make no mis…	
1	It Looks Like Uber's Winning Its War With New …	
2	The Progressive Promise of Today's Technology…	
3	Don't Let These 5 Confusing Words Mar Your Ima…	

```
4  What You Don't Know About Overnight Success. I…

                                    processed_text   topic_0   topic_1  \
0  manage personal brand mistake facebook account…  0.005444  0.005445
1           looks uber winning war york grab popcorn  0.017901  0.017900
2  progressive promise today technology digital p…  0.005004  0.964974
3  let confusing words mar image tom articulate p…  0.006957  0.476840
4  know overnight success fighting thing overnigh…  0.007357  0.007358

    topic_2   topic_3   topic_4   topic_5   topic_6   topic_7  dominant_topic  \
0  0.005446  0.005447  0.005449  0.005440  0.961877  0.005453               6
1  0.017867  0.017866  0.017861  0.017867  0.017859  0.874879               7
2  0.005003  0.005003  0.005004  0.005006  0.005004  0.005002               1
3  0.006960  0.237759  0.006951  0.250616  0.006966  0.006951               1
4  0.007361  0.007360  0.948491  0.007359  0.007357  0.007358               4

   dominant_topic_weight
0               0.961877
1               0.874879
2               0.964974
3               0.476840
4               0.948491
```

1.9.1 9.1. Documentos representativos por tópico

Un documento representativo de un tópico es un documento donde ese tópico tiene un peso alto.

Revisar documentos representativos ayuda a interpretar mejor cada tópico.

[36]:
```python
def mostrar_documentos_representativos(df_resultados, indice_topico,␣
↪numero_documentos=5):
    columna_topico = f"topic_{indice_topico}"

    columnas_mostrar = [
        "category",
        "headline",
        "short_description",
        columna_topico,
        "dominant_topic",
        "dominant_topic_weight"
    ]

    # Ordena los documentos en base a la columna del tópico asociada al índice␣
↪de tópico indicado como parámetro
    documentos = (
        df_resultados
        .sort_values(by=columna_topico, ascending=False)
        .head(numero_documentos)
```

```
    )

    return documentos[columnas_mostrar]

# Ejemplo: documentos más representativos del tópico 0
mostrar_documentos_representativos(df_topicos_documentos, indice_topico=0,␣
 ↪numero_documentos=5)
```

[36]:
```
           category                                              headline  \
846   ENTERTAINMENT  Love Him or Hate Him: Entrepreneurs Can Learn …
204        BUSINESS  Maybe The Federal Reserve Has Been Right About…
3530       WELLNESS                  Does Memory Training Really Work?
1699       POLITICS  States Must Find Way to Do Right by Home Care …
3746       WELLNESS                                      Birth and Death

                                    short_description   topic_0  \
846   Entrepreneurs who are just starting out should…  0.970806
204   Hard as it may be for its legion of economic, …  0.964961
3530  The latest research suggests memory training d…  0.964952
1699  When does justice delayed become justice denie…  0.960181
3746  Lola taught me how to handle adversity and Luc…  0.958258

      dominant_topic  dominant_topic_weight
846                0               0.970806
204                0               0.964961
3530               0               0.964952
1699               0               0.960181
3746               0               0.958258
```

[37]:
```
# Revisamos documentos representativos de todos los tópicos
# Un for es un bucle que repite algo varias veces (en este caso, tantas como␣
 ↪valores para numero_topicos tengamos)
for indice_topico in range(numero_topicos):
    print("=" * 120)
    print(f"TÓPICO {indice_topico}")
    print("- Palabras principales:")
    print(tabla_topicos.loc[tabla_topicos["topic"] == indice_topico,␣
 ↪"top_words"].values[0])
    print("\n- Documentos representativos:")
    display(mostrar_documentos_representativos(df_topicos_documentos,␣
 ↪indice_topico, numero_documentos=3))
```

```
========================================================================================
========================================
TÓPICO 0
- Palabras principales:
trump, donald, president, obama, clinton, house, court, gop, hillary, may,
photos, republican
```

- Documentos representativos:

```
          category                                              headline  \
846   ENTERTAINMENT  Love Him or Hate Him: Entrepreneurs Can Learn …
204        BUSINESS  Maybe The Federal Reserve Has Been Right About…
3530       WELLNESS                  Does Memory Training Really Work?

                                    short_description   topic_0  \
846   Entrepreneurs who are just starting out should…  0.970806
204   Hard as it may be for its legion of economic, …  0.964961
3530  The latest research suggests memory training d…  0.964952

      dominant_topic  dominant_topic_weight
846                0               0.970806
204                0               0.964961
3530               0               0.964952
```

```
================================================================================
========================================
```

TÓPICO 1

- Palabras principales:

first, star, life, way, uber, data, nfl, olympic, sports, night, best, world

- Documentos representativos:

```
      category                                              headline  \
3781  WELLNESS  ACEs: Adverse Childhood Experiences -- and Pro…
2     BUSINESS      The Progressive Promise of Today's Technology
44    BUSINESS                  What Is the Optimal Credit Score?

                                    short_description   topic_1  \
3781  Many youth experience events in their early li…  0.967553
2     A digital policy for the new century, tailored…  0.964974
44    Truthfully, the highest credit score is not th…  0.964951

      dominant_topic  dominant_topic_weight
3781               1               0.967553
2                  1               0.964974
44                 1               0.964951
```

```
================================================================================
========================================
```

TÓPICO 2

- Palabras principales:

city, things, photos, york, best, study, every, back, business, world, getting, google

- Documentos representativos:

```
      category                                           headline  \
3420    TRAVEL  Five Things You Must Buy in the Caribbean -  A…
376   BUSINESS  The Odd Couple In Today's Office:  Millennials…
218   BUSINESS  4 Things They Don't Teach You in MBA Programs …

                                    short_description   topic_2  \
3420  Step away from the two-for-$20 T-shirts, the m…  0.967551
376   The term 'reverse mentor' was coined and first…  0.967547
218   Having spent six years in two different colleg…  0.964956

      dominant_topic  dominant_topic_weight
3420               2               0.967551
376                2               0.967547
218                2               0.964956
```

```
================================================================================
========================================
TÓPICO 3
- Palabras principales:
apple, week, iphone, back, help, best, travel, life, ways, rumors, first,
company

- Documentos representativos:
```

```
           category                                           headline  \
715   ENTERTAINMENT  The Second Best Exotic Marigold Hotel: A Movie…
3088         TRAVEL  5 Secrets To Having a Smarter -- and Happier -…
2460           TECH  App Neutrality Should Be Part of the Net Neutr…

                                    short_description   topic_3  \
715   The return viewer to The Best Exotic Marigold …  0.971728
3088  Money can destroy your trip, but not necessari…  0.967554
2460  I bet you think the mobile Internet is open. T…  0.966294

      dominant_topic  dominant_topic_weight
715                3               0.971728
3088               3               0.967554
2460               3               0.966294
```

```
================================================================================
========================================
TÓPICO 4
- Palabras principales:
food, photos, facebook, think, holiday, well, know, eat, weight, recipes, need,
billion

- Documentos representativos:
```

```
      category                                           headline  \
265   BUSINESS                          Let's Talk Toilet Paper
```

```
3522  WELLNESS    What Would YOU Do With A Second Chance In Life?
3830  WELLNESS  Meditation Tips: Does Meditation Have to Be Se…

                                     short_description   topic_4  \
265   I once worked in a Fortune 500 paper mill that…  0.972627
3522  On Dec. 26, 2004, my life changed forever. Tha…  0.967530
3830  Some people think that their thoughts and feel…  0.964964

      dominant_topic  dominant_topic_weight
265                4               0.972627
3522               4               0.967530
3830               4               0.964964
```

```
================================================================================
=======================================
TÓPICO 5
- Palabras principales:
health, healthy, may, keep, take, want, good, game, cancer, companies, even,
still

- Documentos representativos:
```

```
      category                                          headline  \
1497  POLITICS  Reflections on Justice After Ferguson, MO:  Wh…
3769  WELLNESS  Better Data Will Lead to Better Care for Those…
2153    SPORTS  NFL-Backed Youth Program Says It Reduced Concu…

                                     short_description   topic_5  \
1497  It turns out that justice, like so much of wha…  0.974966
3769  People living with multiple chronic conditions…  0.969798
2153  As increasing numbers of parents keep their ch…  0.963505

      dominant_topic  dominant_topic_weight
1497               5               0.974966
3769               5               0.969798
2153               5               0.963505
```

```
================================================================================
=======================================
TÓPICO 6
- Palabras principales:
world, day, first, state, little, best, play, stress, way, around, video,
facebook

- Documentos representativos:
```

```
           category                                          headline  \
775   ENTERTAINMENT  Filmmaker Fights the Stigma Against Mental Ill…
21         BUSINESS                How Clean Energy Works for Colorado
3872       WELLNESS  Can We Predict How Stressed Out You Are From T…
```

```
                                      short_description   topic_6  \
775   The best thing about the #MyMentalHealthStory …  0.971730
21    Smart clean energy policies like Colorado's pe…  0.966309
3872  We set up a tracking poll in February to gauge…  0.966305

      dominant_topic  dominant_topic_weight
775                6               0.971730
21                 6               0.966309
3872               6               0.966305

================================================================================
=====================================
TÓPICO 7
- Palabras principales:
food, super, day, watch, summer, week, bowl, videos, recipe, video, youtube, see

- Documentos representativos:

          category                                              headline  \
1401  FOOD & DRINK                          9 Labor Day Grill Favorites
1408  FOOD & DRINK                                 Labor Day Cocktails
3039        TRAVEL  Chefs A-Twitter In London, New York And Paris:…

                                      short_description   topic_7  \
1401  For most children, Labor Day is a bittersweet …  0.973444
1408  Labor Day weekend is summer's last long weeken…  0.967566
3039  When planning a trip, I am prejudiced in favor…  0.967545

      dominant_topic  dominant_topic_weight
1401               7               0.973444
1408               7               0.967566
3039               7               0.967545
```

1.10 10. Comparación con las categorías reales

El dataset incluye categorías reales asignadas por HuffPost.

LDA no usa esas categorías durante el entrenamiento. Por eso, esta comparación no es una evaluación supervisada, sino una ayuda interpretativa. No siempre tendremos esta ayuda disponible, dependerá del dataset.

La pregunta que queremos explorar es:

¿los tópicos descubiertos por LDA se parecen a las categorías reales del dataset?

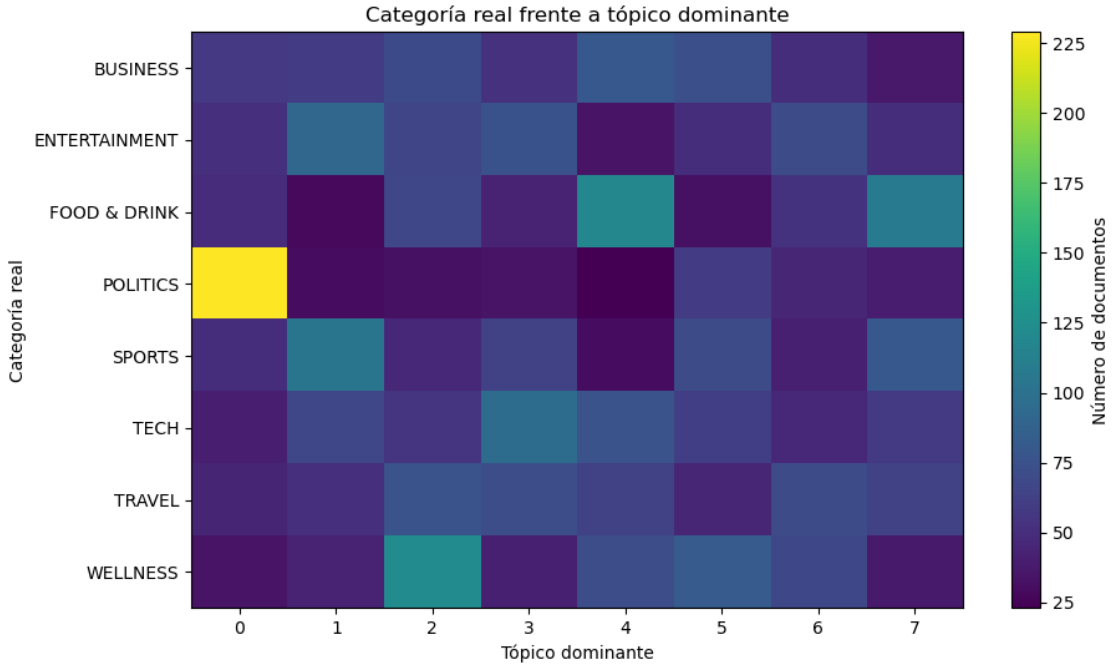
[38]:
```
# Tabla de frecuencia entre categoría real y tópico dominante
# La función crosstab() cruza dos variables categóricas y cuenta cuántos casos␣
 ↪hay en cada combinación
```

```
# En este caso, cuenta cuántos documentos de cada categoría original tienen␣
 ↪cada tópico como tópico dominante
tabla_categoria_topico = pd.crosstab(
    df_topicos_documentos["category"],
    df_topicos_documentos["dominant_topic"]
)

tabla_categoria_topico
```

[38]:

dominant_topic	0	1	2	3	4	5	6	7
category								
BUSINESS	57	59	69	52	80	73	50	37
ENTERTAINMENT	51	91	66	75	34	50	71	50
FOOD & DRINK	49	28	67	43	118	32	53	109
POLITICS	229	30	32	34	23	59	45	39
SPORTS	50	103	46	64	30	71	41	80
TECH	40	67	54	96	76	61	46	58
TRAVEL	44	51	76	72	64	45	70	64
WELLNESS	34	43	122	41	72	82	68	38

[39]:
```
plt.figure(figsize=(10, 6))
plt.imshow(tabla_categoria_topico, aspect="auto")
plt.title("Categoría real frente a tópico dominante")
plt.xlabel("Tópico dominante")
plt.ylabel("Categoría real")
plt.xticks(
    ticks=np.arange(tabla_categoria_topico.shape[1]),
    labels=tabla_categoria_topico.columns
)
plt.yticks(
    ticks=np.arange(tabla_categoria_topico.shape[0]),
    labels=tabla_categoria_topico.index
)
plt.colorbar(label="Número de documentos")
plt.show()
```



1.10.1 10.1. Interpretación de la comparación

La tabla anterior cruza la categoría real del dataset con el tópico dominante de cada documento, es decir, el tópico que tiene mayor peso según LDA.

Es importante recordar que LDA no ha usado la categoría real durante el entrenamiento. La categoría solo se utiliza ahora como referencia para interpretar los resultados.

En los resultados obtenidos se observan varios comportamientos interesantes.

Por un lado, algunas categorías parecen concentrarse con más claridad en determinados tópicos.

Por ejemplo, la categoría `POLITICS` aparece muy concentrada en el tópico 0.
Esto sugiere que el modelo ha aprendido un tópico bastante asociado a política, lo cual encaja con las palabras principales observadas en ese tópico.

También se observa que `FOOD & DRINK` tiene bastante presencia en los tópicos 4 y 7.
Esto puede indicar que el modelo ha dividido el contenido relacionado con comida en más de un tópico, quizá separando recetas, hábitos, eventos o contenido más general.

La categoría `WELLNESS` aparece especialmente asociada al tópico 2, aunque también se reparte entre otros tópicos.
Esto sugiere que el contenido de bienestar puede compartir vocabulario con otros temas, como salud, estilo de vida o alimentación.

Por otro lado, algunas categorías aparecen bastante repartidas entre varios tópicos.

Por ejemplo, `BUSINESS`, `TRAVEL`, `TECH`, `SPORTS` y `ENTERTAINMENT` no se concentran de forma clara en un único tópico.

Esto puede deberse a que sus documentos contienen vocabulario diverso o comparten términos con otras categorías.

Estos resultados muestran una idea importante:

- Una categoría real puede dividirse en varios tópicos.
- Un tópico puede recoger documentos de varias categorías.
- Los tópicos no tienen por qué coincidir exactamente con las categorías del dataset.
- LDA descubre patrones de coocurrencia de palabras, no etiquetas predefinidas.

Por tanto, la comparación con las categorías reales no debe interpretarse como una evaluación exacta de aciertos y errores.

Debe entenderse como una herramienta de apoyo para analizar si los tópicos descubiertos tienen sentido y si capturan parte de la estructura temática del corpus.

1.11 11. Prueba con distintos números de tópicos

El número de tópicos es una decisión importante.

Si usamos pocos tópicos, varios temas pueden mezclarse.

Si usamos demasiados tópicos, un mismo tema puede fragmentarse en varios tópicos parecidos.

A continuación entrenaremos varios modelos con distintos valores de `n_components` para comparar resultados.

[40]:
```
def entrenar_y_resumir_lda(matriz, vocabulario, numero_topicos,␣
↪numero_palabras=8):
    modelo = LatentDirichletAllocation(
        n_components=numero_topicos,
        random_state=42,
        learning_method="batch",
        max_iter=20
    )

    modelo.fit(matriz)

    tabla = obtener_palabras_topico(
        modelo,
        vocabulario,
        numero_palabras=numero_palabras
    )

    return modelo, tabla[["topic", "top_words"]]
```

[41]:
```
valores_k = [4, 6, 8, 10]

resultados_modelos = {}

for k in valores_k:
```

```
    print("=" * 120)
    print(f"Modelo LDA con {k} tópicos")

    modelo_k, tabla_k = entrenar_y_resumir_lda(
        matriz_conteos,
        vocabulario,
        numero_topicos=k,
        numero_palabras=10
    )

    resultados_modelos[k] = {
        "model": modelo_k,
        "topics": tabla_k
    }

    display(tabla_k)
```

```
========================================================================================
========================================
Modelo LDA con 4 tópicos
```

	topic	top_words
0	0	trump, donald, president, food, may, know, oba…
1	1	facebook, star, first, video, world, state, nf…
2	2	life, world, google, day, food, best, business…
3	3	week, apple, day, photos, want, top, watch, ip…

```
========================================================================================
========================================
Modelo LDA con 6 tópicos
```

	topic	top_words
0	0	trump, clinton, better, way, may, know, even, …
1	1	star, first, video, state, night, uber, film, …
2	2	life, world, things, city, back, take, game, y…
3	3	apple, week, iphone, travel, think, best, hous…
4	4	food, photos, day, recipes, best, holiday, wor…
5	5	trump, health, donald, president, even, good, …

```
========================================================================================
========================================
Modelo LDA con 8 tópicos
```

	topic	top_words
0	0	trump, donald, president, obama, clinton, hous…
1	1	first, star, life, way, uber, data, nfl, olymp…
2	2	city, things, photos, york, best, study, every…
3	3	apple, week, iphone, back, help, best, travel,…
4	4	food, photos, facebook, think, holiday, well, …
5	5	health, healthy, may, keep, take, want, good, …

```
6      6  world, day, first, state, little, best, play, …
7      7  food, super, day, watch, summer, week, bowl, v…

================================================================================
========================================
Modelo LDA con 10 tópicos

   topic                                               top_words
0      0  trump, donald, president, live, way, white, fi…
1      1  first, olympic, sports, star, uber, fans, way,…
2      2  things, photos, got, world, google, change, ga…
3      3  travel, apple, nfl, best, think, still, want, …
4      4  know, social, want, facebook, job, news, media…
5      5  health, healthy, take, let, eating, good, keep…
6      6  day, world, twitter, facebook, first, sleep, c…
7      7  food, recipes, photos, summer, week, day, reci…
8      8  women, life, best, world, things, know, photos…
9      9  trump, apple, clinton, court, week, donald, go…
```

[42]:
```
resultados_modelos
```

[42]:
```
{4: {'model': LatentDirichletAllocation(max_iter=20, n_components=4,
random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, donald, president, food, may, know, oba…
  1      1  facebook, star, first, video, world, state, nf…
  2      2  life, world, google, day, food, best, business…
  3      3  week, apple, day, photos, want, top, watch, ip…},
 6: {'model': LatentDirichletAllocation(max_iter=20, n_components=6,
random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, clinton, better, way, may, know, even, …
  1      1  star, first, video, state, night, uber, film, …
  2      2  life, world, things, city, back, take, game, y…
  3      3  apple, week, iphone, travel, think, best, hous…
  4      4  food, photos, day, recipes, best, holiday, wor…
  5      5  trump, health, donald, president, even, good, …},
 8: {'model': LatentDirichletAllocation(max_iter=20, n_components=8,
random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, donald, president, obama, clinton, hous…
  1      1  first, star, life, way, uber, data, nfl, olymp…
  2      2  city, things, photos, york, best, study, every…
  3      3  apple, week, iphone, back, help, best, travel,…
  4      4  food, photos, facebook, think, holiday, well, …
  5      5  health, healthy, may, keep, take, want, good, …
  6      6  world, day, first, state, little, best, play, …
  7      7  food, super, day, watch, summer, week, bowl, v…},
```

```
10: {'model': LatentDirichletAllocation(max_iter=20, random_state=42),
 'topics':    topic                                               top_words
 0      0  trump, donald, president, live, way, white, fi…
 1      1  first, olympic, sports, star, uber, fans, way,…
 2      2  things, photos, got, world, google, change, ga…
 3      3  travel, apple, nfl, best, think, still, want, …
 4      4  know, social, want, facebook, job, news, media…
 5      5  health, healthy, take, let, eating, good, keep…
 6      6  day, world, twitter, facebook, first, sleep, c…
 7      7  food, recipes, photos, summer, week, day, reci…
 8      8  women, life, best, world, things, know, photos…
 9      9  trump, apple, clinton, court, week, donald, go…}}
```

1.11.1 11.1. Comparación de configuraciones

Al comparar distintos números de tópicos, no debemos fijarnos solo en el valor de `k`.

También debemos revisar si los tópicos resultantes son interpretables y útiles para el análisis.

En los resultados obtenidos se observa que:

- Con `k = 4`, los tópicos son demasiado generales. Algunos mezclan palabras de política, comida, tecnología o entretenimiento dentro del mismo tópico.
- Con `k = 6`, empiezan a separarse algunos temas, como comida o tecnología, pero todavía aparecen tópicos mezclados.
- Con `k = 8`, aparecen algunos tópicos más claros, como política, salud o comida, aunque todavía hay tópicos con palabras demasiado generales.
- Con `k = 10`, algunos temas se fragmentan en varios tópicos parecidos. Por ejemplo, aparecen varios tópicos con palabras relacionadas con Trump, política, tecnología o redes sociales.

Esto muestra una idea importante:

- Si usamos pocos tópicos, temas diferentes pueden quedar mezclados.
- Si usamos demasiados tópicos, un mismo tema puede dividirse en varios tópicos similares.
- El mejor valor de `k` no siempre es el que produce más tópicos, sino el que genera tópicos más interpretables y útiles.

En esta práctica, `k = 8` parece una opción razonable como punto de partida, porque permite separar algunos temas relevantes sin fragmentar tanto como `k = 10`.

Aun así, los resultados todavía podrían mejorar ajustando el preprocesamiento, añadiendo stopwords específicas del corpus o modificando los parámetros del vectorizador.

No siempre hay un único valor correcto para el número de tópicos.

1.12 12. Extensión: uso de n-gramas en la representación

En el flujo principal hemos construido la matriz documento-término usando únicamente palabras individuales.

Es decir, cada término del vocabulario era un unigrama.

Sin embargo, algunas expresiones tienen más significado cuando se consideran como secuencias de palabras.

Por ejemplo:

- `donald trump`
- `white house`
- `super bowl`
- `stock market`
- `social media`

Si usamos solo palabras individuales, estas expresiones se separan en términos independientes.

Con `CountVectorizer` podemos ampliar la representación usando el parámetro `ngram_range`.

Por ejemplo:

- `ngram_range=(1, 1)`: usa solo unigramas
- `ngram_range=(1, 2)`: usa unigramas y bigramas
- `ngram_range=(2, 2)`: usa solo bigramas

En esta sección probaremos una representación con unigramas y bigramas para comprobar si los tópicos obtenidos son más interpretables.

Es importante recordar que aquí no estamos construyendo un modelo probabilístico de n-gramas.

Estamos usando n-gramas como términos del vocabulario dentro de la matriz documento-término.

[43]:
```
# Construimos una nueva matriz documento-término usando unigramas y bigramas.
# En este caso, el vocabulario puede contener palabras individuales y pares de␣
 ↪palabras consecutivas.

vectorizador_ngramas = CountVectorizer(
    lowercase=False,
    min_df=5,
    max_df=0.80,
    max_features=3000,
    ngram_range=(1, 2)
)

matriz_ngramas = vectorizador_ngramas.
 ↪fit_transform(df_reducido["processed_text"])
vocabulario_ngramas = vectorizador_ngramas.get_feature_names_out()

print("Dimensiones de la matriz documento-término con unigramas y bigramas:")
print(matriz_ngramas.shape)

print("\nTamaño del vocabulario con n-gramas:", len(vocabulario_ngramas))

print("\nPrimeros términos del vocabulario:")
print(vocabulario_ngramas[:50])
```

```
Dimensiones de la matriz documento-término con unigramas y bigramas:
(3924, 2953)

Tamaño del vocabulario con n-gramas: 2953

Primeros términos del vocabulario:
['abc' 'ability' 'able' 'abortion' 'absolutely' 'abuse' 'academy'
 'accepted' 'access' 'accessible' 'accident' 'according' 'account'
 'accusations' 'accused' 'across' 'across country' 'act' 'acting' 'action'
 'actions' 'active' 'activities' 'activity' 'actor' 'actors' 'actress'
 'actual' 'actually' 'actually want' 'add' 'added' 'adding' 'addition'
 'address' 'adds' 'administration' 'admits' 'admitted' 'adorable' 'ads'
 'adult' 'adults' 'advance' 'advanced' 'advances' 'advantage' 'adventure'
 'advertising' 'advice']
```

1.12.1 12.1. Entrenamiento de LDA con unigramas y bigramas

Una vez construida la matriz documento-término con unigramas y bigramas, podemos entrenar de nuevo LDA.

Para mantener la comparación con el modelo anterior, usaremos inicialmente el mismo número de tópicos.

Reutilizaremos la función `entrenar_y_resumir_lda`, que entrena el modelo y devuelve una tabla con las palabras principales de cada tópico.

[44]:
```
numero_topicos_ngramas = numero_topicos

modelo_lda_ngramas, tabla_topicos_ngramas = entrenar_y_resumir_lda(
    matriz_ngramas,
    vocabulario_ngramas,
    numero_topicos=numero_topicos_ngramas,
    numero_palabras=12
)

tabla_topicos_ngramas
```

[44]:

	topic	top_words
0	0	day, travel, free, around, best, photos, busin…
1	1	day, video, game, women, football, team, nfl, …
2	2	recipes, photos, season, holiday, need, job, u…
3	3	trump, donald, donald trump, health, president…
4	4	food, world, better, twitter, may, life, reall…
5	5	apple, week, food, life, look, back, watch, ta…
6	6	best, want, know, love, find, travel, right, s…
7	7	facebook, social, first, want, olympic, media,…

[45]:
```
print("Tópicos obtenidos con la representación original basada en unigramas:")
display(tabla_topicos[["topic", "top_words"]])
```

```
print("\nTópicos obtenidos con la representación basada en unigramas y bigramas:
  ↪")
display(tabla_topicos_ngramas[["topic", "top_words"]])
```

```
Tópicos obtenidos con la representación original basada en unigramas:
```

	topic	top_words
0	0	trump, donald, president, obama, clinton, hous…
1	1	first, star, life, way, uber, data, nfl, olymp…
2	2	city, things, photos, york, best, study, every…
3	3	apple, week, iphone, back, help, best, travel,…
4	4	food, photos, facebook, think, holiday, well, …
5	5	health, healthy, may, keep, take, want, good, …
6	6	world, day, first, state, little, best, play, …
7	7	food, super, day, watch, summer, week, bowl, v…

```
Tópicos obtenidos con la representación basada en unigramas y bigramas:
```

	topic	top_words
0	0	day, travel, free, around, best, photos, busin…
1	1	day, video, game, women, football, team, nfl, …
2	2	recipes, photos, season, holiday, need, job, u…
3	3	trump, donald, donald trump, health, president…
4	4	food, world, better, twitter, may, life, reall…
5	5	apple, week, food, life, look, back, watch, ta…
6	6	best, want, know, love, find, travel, right, s…
7	7	facebook, social, first, want, olympic, media,…

1.12.2 12.2. Comparación con la representación basada solo en unigramas

Al incluir bigramas, el vocabulario puede contener expresiones de dos palabras que resultan más interpretables que los términos individuales.

En los resultados obtenidos aparecen algunos bigramas informativos, como `donald trump` o `social media`.

Estos términos pueden ayudar a interpretar mejor ciertos tópicos, porque representan expresiones con significado propio.

Sin embargo, la inclusión de bigramas no ha producido una mejora clara en todos los tópicos.

Muchos tópicos siguen estando dominados por palabras individuales demasiado generales, como `day`, `best`, `photos`, `life`, `want`, `know` o `week`.

También se observan tópicos mezclados, donde aparecen términos de distintas áreas temáticas en el mismo grupo.

Esto muestra que añadir bigramas puede ser útil, pero no garantiza automáticamente tópicos más claros.

Su utilidad depende de varios factores:

- La calidad del preprocesamiento
- La lista de stopwords utilizada
- Los valores de `min_df` y `max_df`
- El tamaño del vocabulario permitido
- El número de tópicos seleccionado
- La frecuencia real de las expresiones dentro del corpus

Por tanto, la comparación entre unigramas y unigramas + bigramas debe hacerse revisando la interpretabilidad final de los tópicos.

1.12.3 12.3. Prueba con distintos números de tópicos usando n-gramas

También podemos combinar el uso de n-gramas con la exploración de distintos valores de número de tópicos.

Esto permite comprobar si los bigramas ayudan a obtener tópicos más claros en distintas configuraciones.

[46]:
```
valores_k_ngramas = [4, 6, 8, 10]

resultados_modelos_ngramas = {}

for k in valores_k_ngramas:
    print("=" * 120)
    print(f"Modelo LDA con unigramas y bigramas, k = {k}")

    modelo_k_ngramas, tabla_k_ngramas = entrenar_y_resumir_lda(
        matriz_ngramas,
        vocabulario_ngramas,
        numero_topicos=k,
        numero_palabras=10
    )

    resultados_modelos_ngramas[k] = {
        "model": modelo_k_ngramas,
        "topics": tabla_k_ngramas
    }

    display(tabla_k_ngramas)
```

```
========================================================================================================================
Modelo LDA con unigramas y bigramas, k = 4
```

	topic	top_words
0	0	day, best, know, free, top, recipe, facebook, …
1	1	day, first, world, game, team, women, video, w…
2	2	food, want, season, life, change, ways, think,…
3	3	trump, donald, apple, donald trump, president,…

```
================================================================================
========================================
Modelo LDA con unigramas y bigramas, k = 6

   topic                                          top_words
0      0  facebook, day, google, recipe, health, busines…
1      1  day, social, super, world, video, first, women…
2      2  recipes, want, photos, change, season, life, u…
3      3  trump, donald, donald trump, president, house,…
4      4  photos, world, best, food, city, travel, may, …
5      5  apple, week, love, life, know, look, food, tak…

================================================================================
========================================
Modelo LDA con unigramas y bigramas, k = 8

   topic                                          top_words
0      0  day, travel, free, around, best, photos, busin…
1      1  day, video, game, women, football, team, nfl, …
2      2  recipes, photos, season, holiday, need, job, u…
3      3  trump, donald, donald trump, health, president…
4      4  food, world, better, twitter, may, life, reall…
5      5  apple, week, food, life, look, back, watch, ta…
6      6  best, want, know, love, find, travel, right, s…
7      7  facebook, social, first, want, olympic, media,…

================================================================================
========================================
Modelo LDA con unigramas y bigramas, k = 10

   topic                                          top_words
0      0  day, health, james, old, data, fun, watch, bus…
1      1  women, game, super, team, bowl, first, way, su…
2      2  change, life, american, stop, san, video, goog…
3      3  trump, donald, donald trump, president, clinto…
4      4  world, travel, twitter, city, may, best, reall…
5      5  apple, iphone, week, life, know, good, rumors,…
6      6  facebook, state, right, home, company, best, s…
7      7  first, want, help, social, world, media, show,…
8      8  food, day, photos, recipes, recipe, best, love…
9      9  watch, week, see, videos, know, movie, youtube…
```

[47]:
```
resultados_modelos
```

[47]:
```
{4: {'model': LatentDirichletAllocation(max_iter=20, n_components=4,
random_state=42),
  'topics':    topic                                          top_words
  0      0  trump, donald, president, food, may, know, oba…
  1      1  facebook, star, first, video, world, state, nf…
  2      2  life, world, google, day, food, best, business…
```

```
  3      3  week, apple, day, photos, want, top, watch, ip…},
 6: {'model': LatentDirichletAllocation(max_iter=20, n_components=6,
random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, clinton, better, way, may, know, even, …
  1      1  star, first, video, state, night, uber, film, …
  2      2  life, world, things, city, back, take, game, y…
  3      3  apple, week, iphone, travel, think, best, hous…
  4      4  food, photos, day, recipes, best, holiday, wor…
  5      5  trump, health, donald, president, even, good, …},
 8: {'model': LatentDirichletAllocation(max_iter=20, n_components=8,
random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, donald, president, obama, clinton, hous…
  1      1  first, star, life, way, uber, data, nfl, olymp…
  2      2  city, things, photos, york, best, study, every…
  3      3  apple, week, iphone, back, help, best, travel,…
  4      4  food, photos, facebook, think, holiday, well, …
  5      5  health, healthy, may, keep, take, want, good, …
  6      6  world, day, first, state, little, best, play, …
  7      7  food, super, day, watch, summer, week, bowl, v…},
 10: {'model': LatentDirichletAllocation(max_iter=20, random_state=42),
  'topics':    topic                                               top_words
  0      0  trump, donald, president, live, way, white, fi…
  1      1  first, olympic, sports, star, uber, fans, way,…
  2      2  things, photos, got, world, google, change, ga…
  3      3  travel, apple, nfl, best, think, still, want, …
  4      4  know, social, want, facebook, job, news, media…
  5      5  health, healthy, take, let, eating, good, keep…
  6      6  day, world, twitter, facebook, first, sleep, c…
  7      7  food, recipes, photos, summer, week, day, reci…
  8      8  women, life, best, world, things, know, photos…
  9      9  trump, apple, clinton, court, week, donald, go…}}
```

1.12.4 12.4. Conclusión sobre el uso de n-gramas

El uso de n-gramas puede mejorar la interpretación cuando aparecen expresiones frecuentes con significado propio.

Por ejemplo, bigramas como `donald trump` o `social media` pueden ser más informativos que analizar cada palabra por separado.

Sin embargo, los resultados obtenidos muestran que añadir n-gramas no garantiza automáticamente una mejora clara.

Aunque aparecen algunas expresiones útiles, muchos tópicos siguen dominados por palabras generales como `day`, `world`, `best`, `life`, `photos`, `want`, `know`, `week` o `first`.

También se observa que algunos tópicos continúan mezclando temas diferentes, como política, tecnología, comida, entretenimiento o deporte.

Además, al probar distintos valores de `k`, se mantiene el mismo patrón general:

- Con pocos tópicos, varios temas quedan mezclados
- Con más tópicos, algunos temas empiezan a separarse
- Con demasiados tópicos, ciertos temas se fragmentan o aparecen repetidos en varios tópicos

Esto muestra que los n-gramas son una ampliación posible de la representación, pero no sustituyen a una buena selección del vocabulario ni a una revisión crítica de los resultados.

Al incluir n-gramas, puede ser necesario ajustar otros parámetros:

- `min_df`
- `max_df`
- `max_features`
- `ngram_range`
- Lista de stopwords específicas del corpus
- Número de tópicos

Por tanto, los n-gramas deben evaluarse según si producen tópicos más coherentes e interpretables en el corpus concreto.

En esta práctica, la inclusión de n-gramas aporta algunas expresiones más informativas, pero no resuelve por completo los problemas de mezcla de tópicos y términos demasiado generales.

1.13 13. Limitaciones observadas

Los resultados obtenidos muestran que LDA puede ser muy útil para explorar una colección de documentos, pero también que sus salidas no siempre son limpias o fáciles de interpretar.

En las distintas ejecuciones realizadas hemos observado varios aspectos importantes:

- Algunos tópicos son relativamente claros, como los relacionados con política, salud o comida
- Otros tópicos mezclan términos de varias categorías, como tecnología, entretenimiento, deportes o viajes
- Algunas palabras demasiado generales aparecen en varios tópicos y dificultan la interpretación
- Al cambiar el número de tópicos, los resultados también cambian: con pocos tópicos se mezclan temas, y con demasiados tópicos algunos temas se fragmentan
- Al incluir n-gramas aparecen algunas expresiones útiles, pero no se resuelven automáticamente todos los problemas de interpretación
- La comparación con las categorías reales muestra que un tópico no equivale necesariamente a una categoría del dataset

Estas limitaciones se deben a que LDA depende de varias decisiones tomadas durante el proceso:

- El preprocesamiento aplicado al texto
- La lista de stopwords utilizada
- El filtrado de términos raros o demasiado frecuentes
- El número de tópicos seleccionado
- El uso de unigramas o n-gramas
- El tamaño y la calidad de los documentos
- Las categorías incluidas en el corpus reducido

Por eso, LDA debe entenderse como una herramienta exploratoria.

El modelo ayuda a descubrir patrones de coocurrencia de palabras en grandes colecciones de texto, pero la interpretación final requiere revisión humana.

En la práctica, aplicar LDA suele ser un proceso iterativo: se entrena una primera versión, se revisan los tópicos obtenidos, se ajustan decisiones de preprocesamiento o representación y se comparan de nuevo los resultados.

1.14 14. Extensión opcional: prueba con lematización

En el flujo principal no hemos aplicado lematización para mantener el notebook sencillo y centrado en LDA.

La lematización puede ser útil porque reduce distintas formas de una palabra a una forma base.

Por ejemplo:

- `companies` → `company`,
- `running` → `run`,
- `elections` → `election`.

Esto puede ayudar a que los tópicos sean más limpios. Sin embargo, también añade dependencias y tiempo de procesamiento.

[48]:
```
# Uso de spaCy para lematización.

# En Google Colab puede ser necesario descargar el modelo de inglés.
# Si el modelo ya está disponible, esta línea no hace falta volver a ejecutarla.
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-
any.whl (12.8 MB)
                           12.8/12.8 MB
51.4 MB/s eta 0:00:00 0:00:01
  Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

[49]:
```
import spacy

nlp = spacy.load("en_core_web_sm")

def lematizar_texto(texto):
    """
    Aplica lematización sobre un texto usando spaCy.

    Pasos:
    1. Convierte el texto a minúsculas.
    2. Analiza el texto con el modelo de spaCy.
    3. Conserva únicamente tokens alfabéticos.
    4. Elimina stopwords.
```

```
    5. Conserva lemas con longitud mínima de 3 caracteres.

    Devuelve:
    - Una cadena con los lemas separados por espacios.
    """
    documento = nlp(texto.lower())

    lemas = []

    for token in documento:
        if token.is_alpha and not token.is_stop and len(token.lemma_) >= 3:
            lemas.append(token.lemma_)

    return " ".join(lemas)
```

[50]:
```
# Aplicamos la lematización al texto original.
# Este paso puede tardar un poco, dependiendo del número de documentos.

df_reducido["lemmatized_text"] = df_reducido["text"].apply(lematizar_texto)
df_reducido[["text", "processed_text", "lemmatized_text"]].head()
```

[50]:
```
                                                text  \
0  How to Manage Your Personal Brand. Make no mis…
1  It Looks Like Uber's Winning Its War With New …
2  The Progressive Promise of Today's Technology…
3  Don't Let These 5 Confusing Words Mar Your Ima…
4  What You Don't Know About Overnight Success. I…

                                      processed_text  \
0  manage personal brand mistake facebook account…
1            looks uber winning war york grab popcorn
2  progressive promise today technology digital p…
3  let confusing words mar image tom articulate p…
4  know overnight success fighting thing overnigh…

                                     lemmatized_text
0  manage personal brand mistake facebook account…
1       look like uber win war new york grab popcorn
2  progressive promise today technology digital p…
3  let confusing word mar image tom articulate ph…
4  know overnight success fight thing year overni…
```

[51]:
```
# Es interesante revisar algunas diferencias entre texto preprocesado básico y␣
 ↪lematizado
for indice in range(3):
    print("=" * 100)
    print("Categoría:", df_reducido.loc[indice, "category"])
```

```
    print("\nTexto original:")
    print(df_reducido.loc[indice, "text"])

    print("\nTexto procesado básico:")
    print(df_reducido.loc[indice, "processed_text"])

    print("\nTexto lematizado:")
    print(df_reducido.loc[indice, "lemmatized_text"])
    print()
```

```
================================================================================
====================
Categoría: BUSINESS

Texto original:
How to Manage Your Personal Brand. Make no mistake: If you have a Facebook
account, an Instagram page, a Twitter profile, you are a brand. Every time you
upload a photo, add a link, or post an update, you're putting into the world
another idea of yourself and what you stand for.

Texto procesado básico:
manage personal brand mistake facebook account instagram page twitter profile
brand every upload photo add link post update putting world another idea stand

Texto lematizado:
manage personal brand mistake facebook account instagram page twitter profile
brand time upload photo add link post update put world idea stand

================================================================================
====================
Categoría: BUSINESS

Texto original:
It Looks Like Uber's Winning Its War With New York. Grab the popcorn.

Texto procesado básico:
looks uber winning war york grab popcorn

Texto lematizado:
look like uber win war new york grab popcorn

================================================================================
====================
Categoría: BUSINESS

Texto original:
The Progressive Promise of Today's Technology. A digital policy for the new
```

```
century, tailored not just to the moment but for the future, is vital if we are
to unleash economic growth, shared prosperity, and the full potential of
technology for citizens and consumers. But such a policy architecture requires a
new consensus--on privacy, on security, on customer protections, on growth and
mobility.

Texto procesado básico:
progressive promise today technology digital policy century tailored moment
future vital unleash economic growth shared prosperity full potential technology
citizens consumers policy architecture requires consensus privacy security
customer protections growth mobility

Texto lematizado:
progressive promise today technology digital policy new century tailor moment
future vital unleash economic growth share prosperity potential technology
citizen consumer policy architecture require new consensus privacy security
customer protection growth mobility
```

[52]:
```python
# Construimos una matriz documento-término con conteos usando el texto␣
↪lematizado.

vectorizador_lemas = CountVectorizer(
    lowercase=False,
    min_df=5,
    max_df=0.80,
    max_features=3000
)

matriz_lemas = vectorizador_lemas.fit_transform(df_reducido["lemmatized_text"])

vocabulario_lemas = vectorizador_lemas.get_feature_names_out()

print("Dimensiones de la matriz documento-término lematizada:")
print(matriz_lemas.shape)

print("\nTamaño del vocabulario lematizado:", len(vocabulario_lemas))
print("Primeros términos del vocabulario lematizado:")
print(vocabulario_lemas[:50])
```

```
Dimensiones de la matriz documento-término lematizada:
(3924, 2402)

Tamaño del vocabulario lematizado: 2402
Primeros términos del vocabulario lematizado:
['abandon' 'abc' 'ability' 'able' 'abortion' 'absolutely' 'abuse'
 'academy' 'accept' 'access' 'accessible' 'accident' 'accord' 'account'
 'accusation' 'accuse' 'achieve' 'acknowledge' 'acquire' 'act' 'action'
```

```
 'active' 'activity' 'actor' 'actress' 'actual' 'actually' 'add' 'addict'
 'addition' 'address' 'administration' 'admit' 'adorable' 'adult'
 'advance' 'advanced' 'advantage' 'adventure' 'advertising' 'advice'
 'adviser' 'affair' 'affect' 'afford' 'affordable' 'afraid' 'africa'
 'african' 'aftermath']
```

[53]:
```
valores_k = [4, 6, 8, 10]

resultados_modelos_lemas = {}

for k in valores_k:
    print("=" * 120)
    print(f"Modelo LDA con lematización y {k} tópicos")

    modelo_k_lemas, tabla_k_lemas = entrenar_y_resumir_lda(
        matriz_lemas,
        vocabulario_lemas,
        numero_topicos=k,
        numero_palabras=10
    )

    resultados_modelos_lemas[k] = {
        "model": modelo_k_lemas,
        "topics": tabla_k_lemas
    }

    display(tabla_k_lemas)
```

```
========================================================================================================================
Modelo LDA con lematización y 4 tópicos
```

	topic	top_words
0	0	trump, want, change, way, say, good, know, bre…
1	1	new, year, good, travel, time, day, world, hea…
2	2	day, recipe, photo, video, time, game, olympic…
3	3	new, trump, food, say, people, report, apple, …

```
========================================================================================================================
Modelo LDA con lematización y 6 tópicos
```

	topic	top_words
0	0	trump, change, donald, life, way, say, want, p…
1	1	new, year, time, good, season, life, look, man…
2	2	day, recipe, apple, week, state, house, video,…
3	3	food, new, study, people, like, eat, trump, ca…
4	4	new, photo, travel, good, world, year, know, w…
5	5	facebook, say, company, health, google, day, y…

```
================================================================================
========================================
Modelo LDA con lematización y 8 tópicos

   topic                                          top_words
0      0  want, change, way, time, beer, life, fear, spo…
1      1  job, new, year, video, look, time, watch, week…
2      2  week, stress, day, star, life, player, feel, a…
3      3  food, new, like, study, cancer, win, year, tim…
4      4  photo, new, travel, good, world, year, hotel, …
5      5  facebook, say, google, company, apple, social,…
6      6  trump, say, state, people, president, donald, …
7      7  recipe, day, good, think, health, eat, trump, …

================================================================================
========================================
Modelo LDA con lematización y 10 tópicos

   topic                                          top_words
0      0  way, time, life, change, need, beer, want, bre…
1      1  year, job, season, life, day, time, good, holi…
2      2  day, week, stress, feel, apple, time, star, li…
3      3  new, like, amazon, food, cancer, share, apple,…
4      4  photo, travel, good, new, world, place, hotel,…
5      5  google, new, say, want, apple, year, help, wor…
6      6  trump, people, say, donald, state, company, pr…
7      7  recipe, good, food, think, day, olympic, eat, …
8      8  trump, say, clinton, court, obama, president, …
9      9  video, new, look, week, game, film, super, wat…
```

[54]:
```
resultados_modelos_lemas
```

[54]:
```
{4: {'model': LatentDirichletAllocation(max_iter=20, n_components=4,
random_state=42),
  'topics':    topic                                          top_words
  0      0  trump, want, change, way, say, good, know, bre…
  1      1  new, year, good, travel, time, day, world, hea…
  2      2  day, recipe, photo, video, time, game, olympic…
  3      3  new, trump, food, say, people, report, apple, …},
 6: {'model': LatentDirichletAllocation(max_iter=20, n_components=6,
random_state=42),
  'topics':    topic                                          top_words
  0      0  trump, change, donald, life, way, say, want, p…
  1      1  new, year, time, good, season, life, look, man…
  2      2  day, recipe, apple, week, state, house, video,…
  3      3  food, new, study, people, like, eat, trump, ca…
  4      4  new, photo, travel, good, world, year, know, w…
  5      5  facebook, say, company, health, google, day, y…},
 8: {'model': LatentDirichletAllocation(max_iter=20, n_components=8,
```

```
random_state=42),
  'topics':    topic                                               top_words
  0       0  want, change, way, time, beer, life, fear, spo…
  1       1  job, new, year, video, look, time, watch, week…
  2       2  week, stress, day, star, life, player, feel, a…
  3       3  food, new, like, study, cancer, win, year, tim…
  4       4  photo, new, travel, good, world, year, hotel, …
  5       5  facebook, say, google, company, apple, social,…
  6       6  trump, say, state, people, president, donald, …
  7       7  recipe, day, good, think, health, eat, trump, …},
 10: {'model': LatentDirichletAllocation(max_iter=20, random_state=42),
  'topics':    topic                                               top_words
  0       0  way, time, life, change, need, beer, want, bre…
  1       1  year, job, season, life, day, time, good, holi…
  2       2  day, week, stress, feel, apple, time, star, li…
  3       3  new, like, amazon, food, cancer, share, apple,…
  4       4  photo, travel, good, new, world, place, hotel,…
  5       5  google, new, say, want, apple, year, help, wor…
  6       6  trump, people, say, donald, state, company, pr…
  7       7  recipe, good, food, think, day, olympic, eat, …
  8       8  trump, say, clinton, court, obama, president, …
  9       9  video, new, look, week, game, film, super, wat…}}
```

[55]:
```
# Entrenamos un modelo LDA concreto sobre la versión lematizada para compararlo
# con el modelo principal entrenado sobre processed_text.

numero_topicos_lemas = 8

modelo_lda_lemas = LatentDirichletAllocation(
    n_components=numero_topicos_lemas,
    random_state=42,
    learning_method="batch",
    max_iter=20
)

modelo_lda_lemas.fit(matriz_lemas)

tabla_topicos_lemas = obtener_palabras_topico(
    modelo_lda_lemas,
    vocabulario_lemas,
    numero_palabras=12
)

tabla_topicos_lemas[["topic", "top_words"]]
```

[55]:
```
   topic                                               top_words
0      0  want, change, way, time, beer, life, fear, spo…
```

```
1      1  job, new, year, video, look, time, watch, week…
2      2  week, stress, day, star, life, player, feel, a…
3      3  food, new, like, study, cancer, win, year, tim…
4      4  photo, new, travel, good, world, year, hotel, …
5      5  facebook, say, google, company, apple, social,…
6      6  trump, say, state, people, president, donald, …
7      7  recipe, day, good, think, health, eat, trump, …
```

1.14.1 Conclusión sobre la lematización

Tras aplicar lematización y volver a entrenar LDA, no se observa una mejora clara en la interpretabilidad de los tópicos.

Aunque la lematización reduce variantes de una misma palabra, los tópicos obtenidos siguen mostrando varios problemas:

- Aparecen palabras demasiado generales en distintos tópicos
- Algunos tópicos siguen mezclando temas diferentes
- La separación entre categorías no mejora de forma evidente

Esto muestra una idea importante: aplicar más preprocesamiento no garantiza automáticamente mejores resultados.

La utilidad de una técnica como la lematización debe evaluarse en función del resultado final.

En el caso de LDA, lo importante no es solo reducir palabras a su forma base, sino obtener tópicos coherentes, interpretables y útiles para el análisis.

Por tanto, en esta práctica mantendremos como flujo principal el modelo entrenado con el preprocesamiento básico, y consideraremos la lematización como una posibilidad más que podría explorarse.

1.15 15. Extensión: representación con TF-IDF

TF-IDF es una representación muy útil en NLP, pero en este notebook no la hemos usado como entrada principal de LDA.

La razón es que LDA clásico se interpreta mejor usando conteos de palabras, porque intenta explicar ocurrencias observadas de términos.

Aun así, TF-IDF es muy útil para otras tareas:

- Identificar términos distintivos de un documento
- Comparar documentos
- Recuperar documentos similares
- Entrenar clasificadores clásicos
- Hacer clustering documental

En esta extensión construiremos una matriz TF-IDF para compararla con la matriz de conteos.

[56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizador_tfidf = TfidfVectorizer(
    lowercase=False,
```

```
    min_df=5,
    max_df=0.80,
    max_features=3000
)

matriz_tfidf = vectorizador_tfidf.fit_transform(df_reducido["processed_text"])
vocabulario_tfidf = vectorizador_tfidf.get_feature_names_out()

# Las dimensiones son las mismas que para la matriz documento-término. La␣
 ↪diferencia
# está en que con TF-IDF cada celda tiene la importancia relativa (peso) de un␣
 ↪término
# en un documento dentro del corpus
print("Dimensiones de la matriz TF-IDF:")
print(matriz_tfidf.shape)
```

```
Dimensiones de la matriz TF-IDF:
(3924, 2797)
```

1.15.1 15.1. Términos con mayor TF-IDF en un documento

A continuación observamos qué términos tienen mayor peso TF-IDF en un documento concreto.

Esto nos ayuda a identificar palabras distintivas del documento.

[57]:
```
indice_documento = 0

vector_tfidf_documento = matriz_tfidf[indice_documento].toarray()[0]

terminos_tfidf = []

for indice_termino, valor_tfidf in enumerate(vector_tfidf_documento):
    if valor_tfidf > 0:
        terminos_tfidf.append((vocabulario_tfidf[indice_termino], valor_tfidf))

df_tfidf_documento = pd.DataFrame(
    terminos_tfidf,
    columns=["term", "tfidf"]
).sort_values(by="tfidf", ascending=False)

print("Categoría real:", df_reducido.loc[indice_documento, "category"])
print("Texto original:")
print(df_reducido.loc[indice_documento, "text"])
print("\nTérminos con mayor TF-IDF:")
df_tfidf_documento.head(15)
```

```
Categoría real: BUSINESS
Texto original:
How to Manage Your Personal Brand. Make no mistake: If you have a Facebook
```

```
account, an Instagram page, a Twitter profile, you are a brand. Every time you
upload a photo, add a link, or post an update, you're putting into the world
another idea of yourself and what you stand for.

Términos con mayor TF-IDF:
```

[57]:

	term	tfidf
3	brand	0.436596
8	link	0.238437
10	mistake	0.234093
9	manage	0.234093
15	profile	0.230261
0	account	0.230261
11	page	0.223733
19	update	0.218298
16	putting	0.213643
1	add	0.207711
6	idea	0.198352
17	stand	0.197024
7	instagram	0.195748
14	post	0.195748
12	personal	0.191093

1.15.2 15.2. Similitud entre documentos usando TF-IDF

Como extensión, podemos calcular similitud entre documentos a partir de sus vectores TF-IDF.

[58]:
```python
from sklearn.metrics.pairwise import cosine_similarity

indice_consulta = 0

similitudes = cosine_similarity(
    matriz_tfidf[indice_consulta],
    matriz_tfidf
).flatten()

# Ordenamos documentos por similitud descendente.
# Excluimos el propio documento, que tendrá similitud 1 consigo mismo.
indices_ordenados = similitudes.argsort()[::-1]

documentos_similares = []

for indice in indices_ordenados:
    if indice != indice_consulta:
        documentos_similares.append(indice)
    if len(documentos_similares) == 5:
        break
```

```python
print("Documento de consulta:")
print("Categoría:", df_reducido.loc[indice_consulta, "category"])
print(df_reducido.loc[indice_consulta, "text"])

print("\nDocumentos más similares según TF-IDF:")
for indice in documentos_similares:
    print("=" * 100)
    print("Similitud:", similitudes[indice])
    print("Categoría:", df_reducido.loc[indice, "category"])
    print(df_reducido.loc[indice, "text"])
```

```
Documento de consulta:
Categoría: BUSINESS
How to Manage Your Personal Brand. Make no mistake: If you have a Facebook
account, an Instagram page, a Twitter profile, you are a brand. Every time you
upload a photo, add a link, or post an update, you're putting into the world
another idea of yourself and what you stand for.

Documentos más similares según TF-IDF:
====================================================================================
====================
Similitud: 0.2693665162274382
Categoría: TECH
Twitter, Instagram Deal Almost Happened For $525 Million: NYT. Relations between
Twitter, Instagram and Facebook have soured since Facebook successfully swooped
for the photo service. Earlier
====================================================================================
====================
Similitud: 0.21638978927152627
Categoría: TECH
Facebook Announces Animated Profile Pics. Oh boy.
====================================================================================
====================
Similitud: 0.19812405766694044
Categoría: TECH
Brace Yourself For Even More Facebook Notifications. Here's the latest update
that Facebook hopes will suck you in.
====================================================================================
====================
Similitud: 0.1613125279881371
Categoría: TECH
How Joining Facebook Is Hurting Instagram. Just months ago Instagram celebrated
a billion-dollar buyout. Now it's falling along with Facebook. Read more on The
Daily
====================================================================================
====================
Similitud: 0.16061489365382442
Categoría: POLITICS
```

```
'Better Late Than Never': Account Reposting Trump Tweets Verbatim Reacts To
Twitter Ban. The @SuspendThePres account reposted Trump's words for months to
prove that an average citizen would be banned for such rhetoric.
```

1.16 16. Conclusiones

En este notebook hemos construido un flujo completo de procesamiento y detección de tópicos sobre un corpus de noticias.

Los pasos principales han sido:

- Cargar el dataset de noticias
- Explorar las categorías disponibles
- Seleccionar un subconjunto de categorías para la práctica
- Construir un texto de trabajo combinando titular y descripción
- Aplicar un preprocesamiento básico sobre los textos
- Crear una matriz documento-término con conteos
- Entrenar un modelo LDA
- Interpretar los tópicos aprendidos a partir de sus palabras principales
- Analizar documentos representativos de cada tópico
- Comparar los tópicos obtenidos con las categorías reales del dataset
- Probar distintos números de tópicos
- Revisar extensiones con lematización y TF-IDF

La idea más importante es que LDA no devuelve etiquetas cerradas.

El modelo devuelve distribuciones de palabras para cada tópico y distribuciones de tópicos para cada documento.

Por tanto, el análisis de tópicos combina modelado automático e interpretación humana.

Además, los resultados dependen de decisiones como el preprocesamiento, el vocabulario, el número de tópicos y los parámetros utilizados.

Por eso, LDA debe entenderse como una herramienta exploratoria que ayuda a descubrir patrones temáticos, pero cuyos resultados deben revisarse e interpretarse cuidadosamente.