Laboratorio 5. Clasificación de tweets usando mineria de texto

integrantes:

- Francis Aguilar - 22243
- Angela García -22869
- Cesar Lopez - 22535

enlace al repositorio: https://github.com/angelargd8/lab5-ds

In [2]:
```
!pip install nltk
!pip install wordcloud
!pip install emoji
```

```
Collecting nltk
  Using cached nltk-3.9.1-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: click in c:\users\msi\desktop\info\escritorio\uvg\8vo
semestre\data science\dsvenv\lib\site-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in c:\users\msi\desktop\info\escritorio\uvg\8v
o semestre\data science\dsvenv\lib\site-packages (from nltk) (1.5.1)
Collecting regex>=2021.8.3 (from nltk)
  Using cached regex-2025.7.34-cp313-cp313-win_amd64.whl.metadata (41 kB)
Requirement already satisfied: tqdm in c:\users\msi\desktop\info\escritorio\uvg\8vo
semestre\data science\dsvenv\lib\site-packages (from nltk) (4.67.1)
Requirement already satisfied: colorama in c:\users\msi\desktop\info\escritorio\uvg
\8vo semestre\data science\dsvenv\lib\site-packages (from click->nltk) (0.4.6)
Using cached nltk-3.9.1-py3-none-any.whl (1.5 MB)
Using cached regex-2025.7.34-cp313-cp313-win_amd64.whl (275 kB)
Installing collected packages: regex, nltk

   ---------------------------------------- 0/2 [regex]
   ---------------------------------------- 0/2 [regex]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
```

```
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   -------------------- ------------------- 1/2 [nltk]
   ---------------------------------------- 2/2 [nltk]

Successfully installed nltk-3.9.1 regex-2025.7.34
```

```
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Collecting wordcloud
  Downloading wordcloud-1.9.4-cp313-cp313-win_amd64.whl.metadata (3.5 kB)
Requirement already satisfied: numpy>=1.6.1 in c:\users\msi\desktop\info\escritorio
\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from wordcloud) (2.3.1)
Requirement already satisfied: pillow in c:\users\msi\desktop\info\escritorio\uvg\8v
o semestre\data science\dsvenv\lib\site-packages (from wordcloud) (11.3.0)
Requirement already satisfied: matplotlib in c:\users\msi\desktop\info\escritorio\uv
g\8vo semestre\data science\dsvenv\lib\site-packages (from wordcloud) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\msi\desktop\info\escrito
rio\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordclo
ud) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\msi\desktop\info\escritorio
\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordcloud)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\msi\desktop\info\escrit
orio\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordcl
oud) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\msi\desktop\info\escrit
orio\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordcl
oud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\msi\desktop\info\escritor
io\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordclou
d) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\msi\desktop\info\escrito
rio\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wordclo
ud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\msi\desktop\info\esc
ritorio\uvg\8vo semestre\data science\dsvenv\lib\site-packages (from matplotlib->wor
dcloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\msi\desktop\info\escritorio\uvg
\8vo semestre\data science\dsvenv\lib\site-packages (from python-dateutil>=2.7->matp
lotlib->wordcloud) (1.17.0)
Downloading wordcloud-1.9.4-cp313-cp313-win_amd64.whl (300 kB)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.4
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
Collecting emoji
  Downloading emoji-2.14.1-py3-none-any.whl.metadata (5.7 kB)
Downloading emoji-2.14.1-py3-none-any.whl (590 kB)
   ---------------------------------------- 0.0/590.6 kB ? eta -:--:--
   ---------------- ---------------------- 262.1/590.6 kB ? eta -:--:--
   ---------------------------------------- 590.6/590.6 kB 4.0 MB/s eta 0:00:00
Installing collected packages: emoji
Successfully installed emoji-2.14.1
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:

```python
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
```

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score
```

In [4]:
```
# Descargar stopwords de NLTK
nltk.download("stopwords")
stop_words   = set(stopwords.words("english"))
stop_words
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\MSI\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[4]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 'couldn',
 "couldn't",
 'd',
 'did',
 'didn',
 "didn't",
 'do',
 'does',
 'doesn',
 "doesn't",
 'doing',
 'don',
 "don't",
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 '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',
                'll',
                'm',
                'ma',
                'me',
                'mightn',
                "mightn't",
                'more',
                'most',
                'mustn',
                "mustn't",
                'my',
                'myself',
                'needn',
                "needn't",
                'no',
                'nor',
                'not',
                'now',
                'o',
                'of',
                'off',
                'on',
                'once',
                'only',
```

```
                    'or',
                    'other',
                    'our',
                    'ours',
                    'ourselves',
                    'out',
                    'over',
                    'own',
                    're',
                    's',
                    'same',
                    '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',
                    'to',
                    'too',
                    '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",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

Dataset de tweets

El conjunto de datos está formado por más de 10 500 filas y 5 columnas:

id: El identificador del tweet

keyword: una palabra clave del tweet, puede estar en blanco

location: la ubicación desde donde fue enviado el tweet

text: El texto del tweet

target: La etiqueta de clasificación que especifica si el tweet se trata de un desastre real (1) o no (0).

In [5]:
```
df = pd.read_csv("train.csv")
df.head()
```

Out[5]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

Analisis exploratorio

In [6]:
```
df.describe()
```

Out[6]:

	id	target
count	7613.000000	7613.00000
mean	5441.934848	0.42966
std	3137.116090	0.49506
min	1.000000	0.00000
25%	2734.000000	0.00000
50%	5408.000000	0.00000
75%	8146.000000	1.00000
max	10873.000000	1.00000

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        7613 non-null   int64 
 1   keyword   7552 non-null   object
 2   location  5080 non-null   object
 3   text      7613 non-null   object
 4   target    7613 non-null   int64 
dtypes: int64(2), object(3)
memory usage: 297.5+ KB
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['id', 'keyword', 'location', 'text', 'target'], dtype='object')
```

In [9]:
```
df.value_counts()
```

Out[9]:
```
id     keyword  location                        text
target
48     ablaze   Birmingham                      @bbcmtd Wholesale Markets ablaze ht
tp://t.co/lHYXEOHY6C                                                           1
1
49     ablaze   Est. September 2012 - Bristol  We always try to bring the heavy. #
metal #RT http://t.co/YAo1e0xngw                                               0
1
50     ablaze   AFRICA                          #AFRICANBAZE: Breaking news:Nigeria
flag set ablaze in Aba. http://t.co/2nndBGwyEi                                1
1
52     ablaze   Philadelphia, PA                Crying out for more! Set me ablaze
0         1
53     ablaze   London, UK                      On plus side LOOK AT THE SKY LAST N
IGHT IT WAS ABLAZE http://t.co/qqsmshaJ3N                                      0
1

..
10826  wrecked  TN                              On the bright side I wrecked htt
p://t.co/uEa0txRHYs                                                               0
1
10829  wrecked  #NewcastleuponTyne #UK          @widda16 ... He's gone. You can rel
ax. I thought the wife who wrecked her cake was a goner mind lol #whoops  0
1
10831  wrecked  Vancouver, Canada               Three days off from work and they'v
e pretty much all been wrecked hahaha shoutout to my family for that one  0
1
10832  wrecked  London                          #FX #forex #trading Cramer: Iger's
3 words that wrecked Disney's stock http://t.co/7enNulLKzM                0
1
10833  wrecked  Lincoln                         @engineshed Great atmosphere at the
British Lion gig tonight. Hearing is wrecked. http://t.co/oMNBAtJEAO     0
1
Name: count, Length: 5080, dtype: int64
```

In [10]:
```
#contar valores duplicadosss
df.duplicated().sum()
```

Out[10]:
```
np.int64(0)
```

In [11]:
```
#contar valores nulos
df.isnull().sum()
```

Out[11]:
```
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
```

In [12]:
```
# pass si es keyword nulo, porque es opcional, agregarle un valor por defecto
df["keyword"].fillna("sin keyword", inplace=True)

# colocar no location
df["location"].fillna("sin location", inplace=True)
```

```
C:\Users\MSI\AppData\Local\Temp\ipykernel_19552\2483260786.py:2: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignmen
t using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["keyword"].fillna("sin keyword", inplace=True)
C:\Users\MSI\AppData\Local\Temp\ipykernel_19552\2483260786.py:5: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignmen
t using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["location"].fillna("sin location", inplace=True)
```

In [13]:
```
#tamanio del dataset
df.shape
```

Out[13]:
```
(7613, 5)
```

In [14]:
```
#tablas de contingencia
#frecuencias absolutas
pd.crosstab(df["keyword"], df["target"]).head(10)
```

Out[14]:

target	0	1
keyword		
ablaze	23	13
accident	11	24
aftershock	34	0
airplane%20accident	5	30
ambulance	18	20
annihilated	23	11
annihilation	19	10
apocalypse	23	9
armageddon	37	5
army	29	5

El resultado es una tabla con conteos por conbinacion de keyword y target

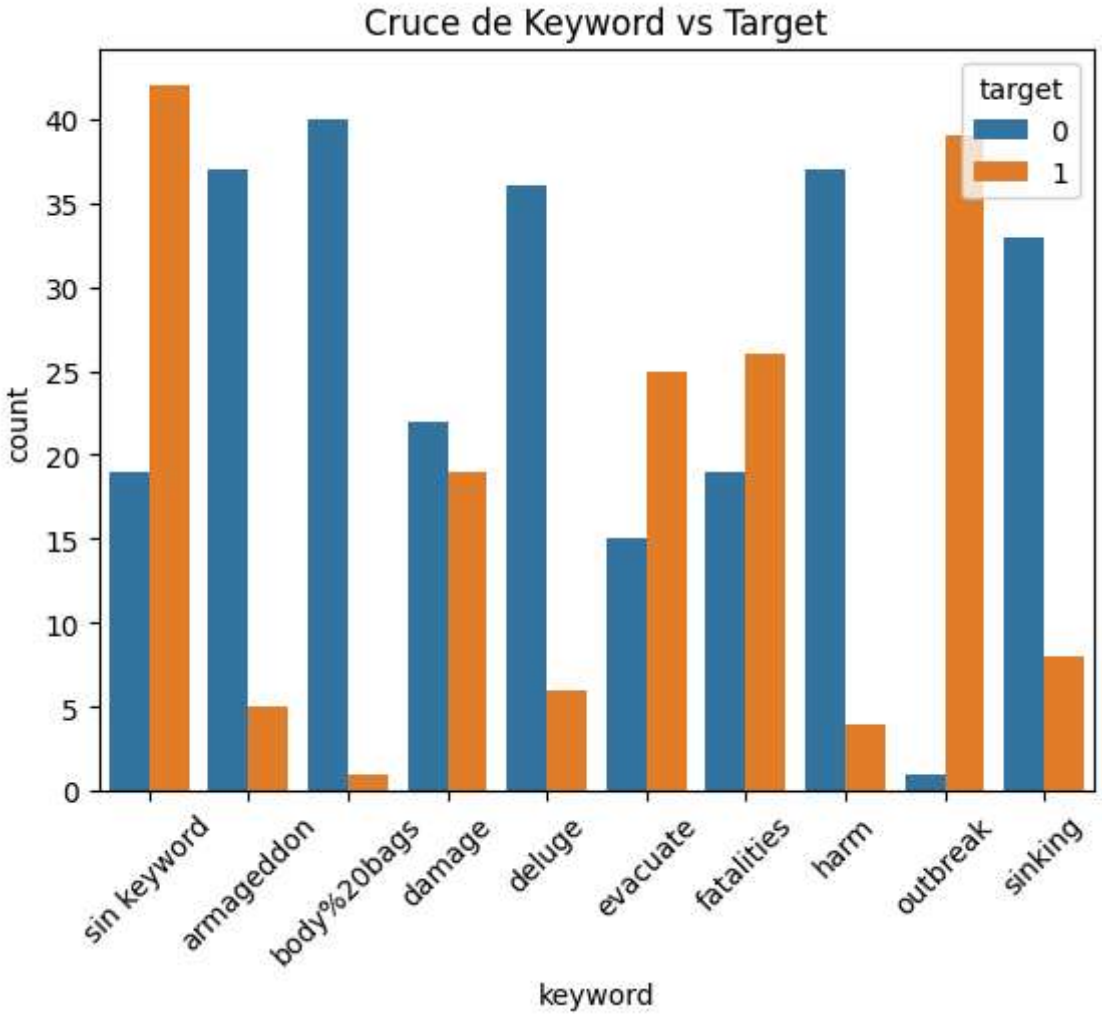
In [15]:
```
pd.crosstab(df["keyword"], df["target"], normalize="index").head(10)
```

Out[15]:

target	0	1
keyword		
ablaze	0.638889	0.361111
accident	0.314286	0.685714
aftershock	1.000000	0.000000
airplane%20accident	0.142857	0.857143
ambulance	0.473684	0.526316
annihilated	0.676471	0.323529
annihilation	0.655172	0.344828
apocalypse	0.718750	0.281250
armageddon	0.880952	0.119048
army	0.852941	0.147059

el resultado de esta tabla de proporcion es que para keyword, que proporcion pertenece a target 0 vs target 1

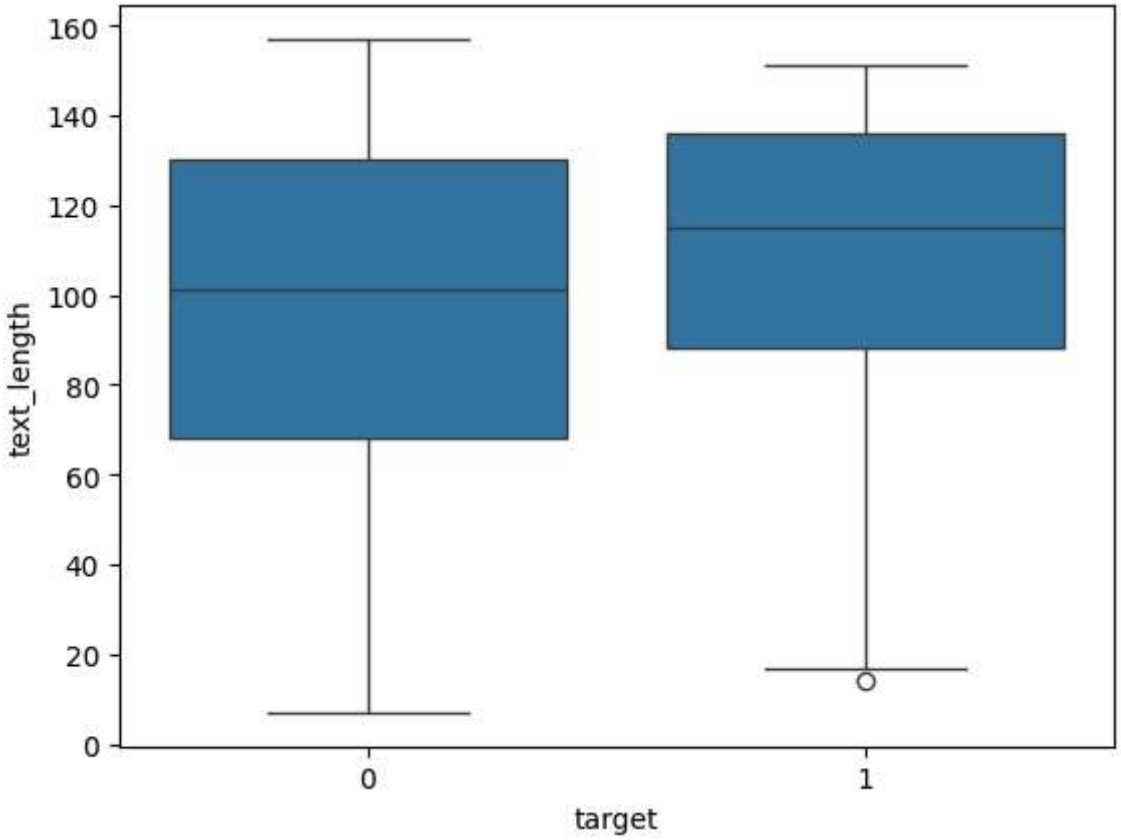
In [16]:
```
import seaborn as sns

top_keywords = df["keyword"].value_counts().head(10).index
sns.countplot(data=df[df["keyword"].isin(top_keywords)],
              x="keyword", hue="target")
plt.xticks(rotation=45)
plt.title("Cruce de Keyword vs Target")
plt.show()
```



Lo que se logra observar es la propocion para pertenece a target 0 o 1. Siendo, la target si el tweet trata de un desastre real cuando es 1 y 0 cuando no lo es. Para cada palabra se nota una gran diferencia de distribución. Y las palabras como armageddon, deluge, harm la mayoria de tweets están en contextos que no son de desastres reales, podrían ser relacionados con contextos humoristicos, ironía o metafóricos. Mientras, las keywords más ligadas a los desastres faltales son evacuate, fatalities y outbreak lo que puede que estan más relacionadas a reportes de emergencias reales. Y en el caso de los que no tienen keyword, la mayoría parece que están relacionados con desastres reales, pero no es suficiente por dí solo para distinguir desastres.

In [17]:
```
#cruce con variables de texto
df["text_length"] = df["text"].apply(len)
sns.boxplot(x="target", y="text_length", data=df)
```

Out[17]:
```
<Axes: xlabel='target', ylabel='text_length'>
```



En el gráfico de cruce con variables de texto de longitud de texto según la variable, lo que se puede observar es que la media de los tweets que no son de desastre tienen alrededor de una media de 50 caracteres. Mientras, los tweets con target 1, que indican un desastre real tienen una media más alta de 60 cáracteres, lo que puede indicar que los que describen un desastre real suelen ser más largos. Esto puede ser debido a que los tweets de desastres reales tienden a ser más largo porque den contexto como del lugar daños o personas.

3. limpieza y preprocesamiento de los datos

In [18]:
```
import emoji
import string
import unicodedata

columnas_cat = df.select_dtypes(include=['object']).columns

important_numbers = {"911", "112", "999", "420", "19"}

def clean_text(text):

    text = text.lower()                                  # minúsculas
    text = re.sub(r"http\S+|www\S+|https\S+", "", text) # quitar URLs
    text = re.sub(r"@\w+", "", text)                     # quitar menciones

    text = re.sub(r"#[\w-]+", "", text)                  # quitar hashtags

    text = re.sub(r"\d+", "", text)                      # quitar números
```

```
    text = re.sub(r"'\w+", "", text)                        # quitar apostrofes

    text = re.sub(r"http\S+|www\S+|https\S+", "", text) # quitar URLs

    text = emoji.replace_emoji(text, replace="") #quitar emojis, al parecer tambien

    #fechas 12/05/2025, 2025-05-12
    text = re.sub(r"\b\d{1,2}[/-]\d{1,2}[/-]\d{2,4}\b", "<date>", text)

    # Horas  12:30, 23:59
    text = re.sub(r"\b\d{1,2}:\d{2}\b", "<time>", text)

    # Cantidades con K, M, % ( 100k, 5%, 2m)
    text = re.sub(r"\b\d+(k|m|%)\b", "<qty>", text)

    # Numeros en general: si es importante, mantenerlo si no, <num>
    def replace_numbers(match):
        num = match.group()
        return num if num in important_numbers else "<num>"

    text = re.sub(r"\b\d+\b", replace_numbers, text)

    #quitar acentos
    text = unicodedata.normalize("NFKD", text).encode("ASCII", "ignore").decode("ut

    text = text.translate(str.maketrans("", "", string.punctuation))  # quitar punt
    text = re.sub(r"\s+", " ", text).strip()  # quitar espacios extra

    #tokenizar
    tokens = text.split()
    #quitar stopwords
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)

df[columnas_cat] = df[columnas_cat].applymap(clean_text)
df
```

```
C:\Users\MSI\AppData\Local\Temp\ipykernel_19552\2711116691.py:53: FutureWarning: Dat
aFrame.applymap has been deprecated. Use DataFrame.map instead.
  df[columnas_cat] = df[columnas_cat].applymap(clean_text)
```

Out[18]:

	id	keyword	location	text	target	text_length
0	1	sin keyword	sin location	deeds reason may allah forgive us	1	69
1	4	sin keyword	sin location	forest fire near la ronge sask canada	1	38
2	5	sin keyword	sin location	residents asked place notified officers evacua...	1	133
3	6	sin keyword	sin location	people receive evacuation orders california	1	65
4	7	sin keyword	sin location	got sent photo ruby smoke pours school	1	88
...	...	...	...	...	...	...
7608	10869	sin keyword	sin location	two giant cranes holding bridge collapse nearb...	1	83
7609	10870	sin keyword	sin location	control wild fires california even northern pa...	1	125
7610	10871	sin keyword	sin location	utckm volcano hawaii	1	65
7611	10872	sin keyword	sin location	police investigating ebike collided car little...	1	137
7612	10873	sin keyword	sin location	latest homes razed northern california wildfir...	1	94

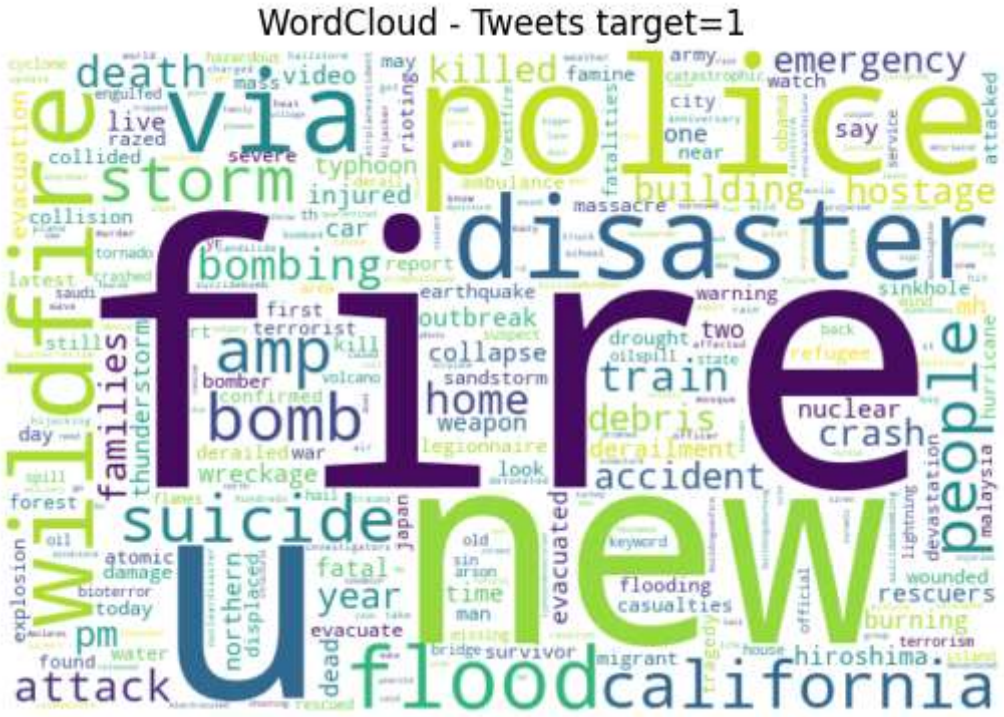
7613 rows × 6 columns

Análisis Exploratorio de Texto 🔍

1. Revisemos las palabras más comunes en target = 1 y 0
2. Visualicemos estas palabras en **nubes de palabras** para entender mejor los patrones.

In [19]:
```
# Texto combinado por clase
cols_wc = ["text", "keyword"]  # sin location
target1_words = df.loc[df.target==1, cols_wc].fillna("").agg(" ".join, axis=1).str.
target2_words = df.loc[df.target==0, cols_wc].fillna("").agg(" ".join, axis=1).str.
```

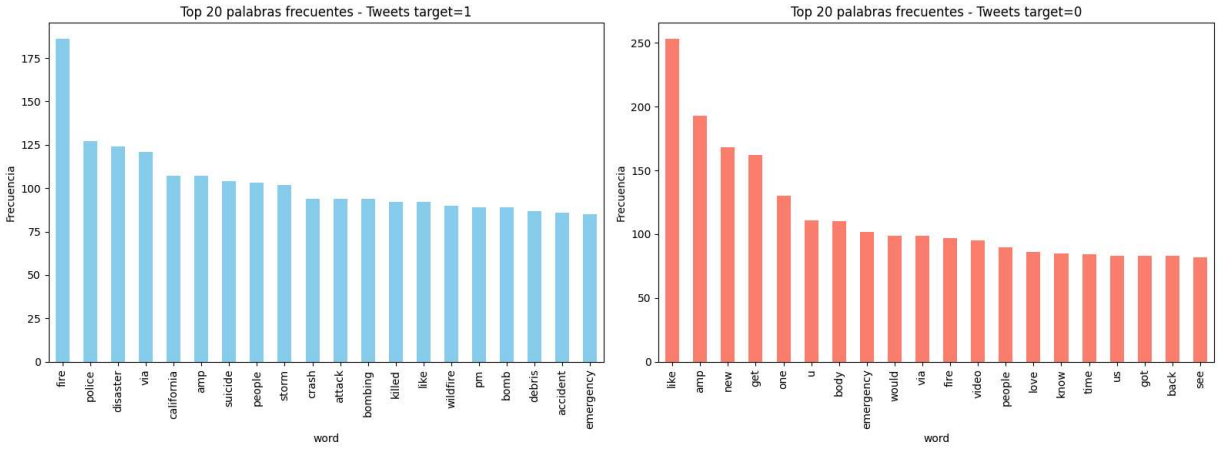
In [20]:
```
wc_params = dict(width=600, height=400, background_color="white",
                 collocations=False, max_words=300)

wordcloud_pos = WordCloud(**wc_params).generate(target1_words)
plt.imshow(wordcloud_pos, interpolation="bilinear")
plt.axis("off")
plt.title("WordCloud - Tweets target=1")
plt.show()
```



In [27]:
```
wordcloud_neg = WordCloud(**wc_params, colormap="Reds").generate(target2_words)
plt.imshow(wordcloud_neg, interpolation="bilinear")
plt.axis("off")
plt.title("WordCloud - Tweets target=0")
plt.show()
```

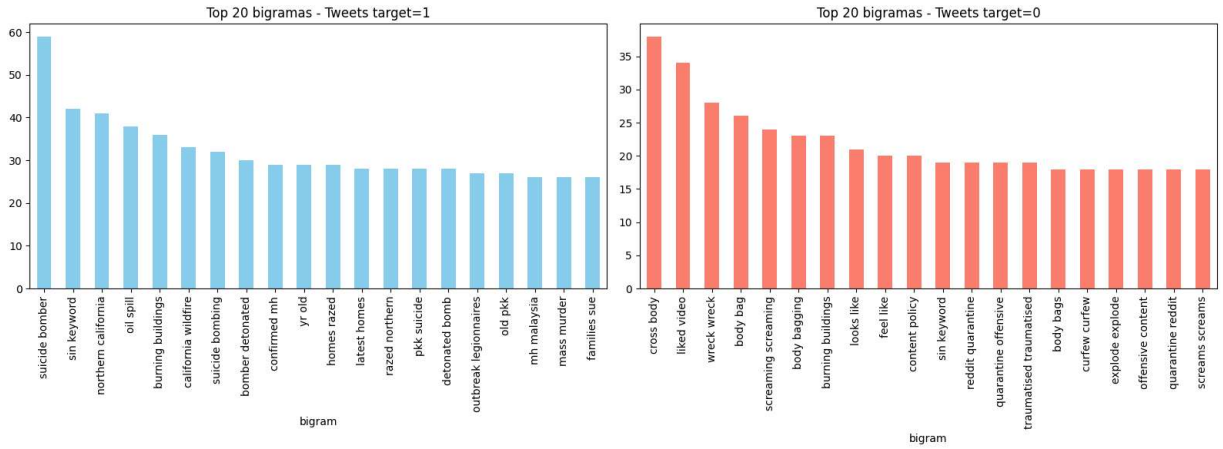
In [31]:
```
# Histogramas de frecuencias palabras individuales
from collections import Counter
import pandas as pd
```

```
target1_counts = Counter(target1_words.split())
target0_counts = Counter(target2_words.split())

df_target1 = pd.DataFrame(target1_counts.most_common(20), columns=['word', 'count']
df_target0 = pd.DataFrame(target0_counts.most_common(20), columns=['word', 'count']

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

df_target1.plot.bar(x='word', y='count', ax=axes[0], color="skyblue", legend=False)
axes[0].set_title("Top 20 palabras frecuentes - Tweets target=1")
axes[0].set_ylabel("Frecuencia")

df_target0.plot.bar(x='word', y='count', ax=axes[1], color="salmon", legend=False)
axes[1].set_title("Top 20 palabras frecuentes - Tweets target=0")
axes[1].set_ylabel("Frecuencia")

plt.tight_layout()
plt.show()
```



In [32]:
```
# Histogramas de frecuencias duplas de palabras

vectorizer = CountVectorizer(ngram_range=(2,2), stop_words='english')

# Target 1
X1 = vectorizer.fit_transform([target1_words])
sum_words1 = X1.toarray().sum(axis=0)
words_freq1 = [(word, sum_words1[idx]) for word, idx in vectorizer.vocabulary_.item
df_bigram1 = pd.DataFrame(sorted(words_freq1, key=lambda x: x[1], reverse=True)[:20
                          columns=['bigram','count'])

# Target 0
X0 = vectorizer.fit_transform([target2_words])
sum_words0 = X0.toarray().sum(axis=0)
words_freq0 = [(word, sum_words0[idx]) for word, idx in vectorizer.vocabulary_.item
df_bigram0 = pd.DataFrame(sorted(words_freq0, key=lambda x: x[1], reverse=True)[:20
                          columns=['bigram','count'])

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
df_bigram1.plot.bar(x='bigram', y='count', ax=axes[0], color="skyblue", legend=Fals
axes[0].set_title("Top 20 bigramas - Tweets target=1")

df_bigram0.plot.bar(x='bigram', y='count', ax=axes[1], color="salmon", legend=False
```

```
axes[1].set_title("Top 20 bigramas - Tweets target=0")

plt.tight_layout()
plt.show()
```



In [33]:
```
df[df["text"].str.contains("wreck wreck", case=False, na=False)]
```

Out[33]:

	id	keyword	location	text	target	text_length
7472	10689	wreck	sin location	wreck wreck wreck wreck wreck wreck wreck wrec...	0	79
7487	10709	wreck	sin location	emotions train wreck body train wreck wreck	0	68

Creacion de modelo de prediccion

In [22]:
```
df
```

Out[22]:

	id	keyword	location	text	target	text_length
0	1	sin keyword	sin location	deeds reason may allah forgive us	1	69
1	4	sin keyword	sin location	forest fire near la ronge sask canada	1	38
2	5	sin keyword	sin location	residents asked place notified officers evacua...	1	133
3	6	sin keyword	sin location	people receive evacuation orders california	1	65
4	7	sin keyword	sin location	got sent photo ruby smoke pours school	1	88
...	...	...	...	...	...	...
7608	10869	sin keyword	sin location	two giant cranes holding bridge collapse nearb...	1	83
7609	10870	sin keyword	sin location	control wild fires california even northern pa...	1	125
7610	10871	sin keyword	sin location	utckm volcano hawaii	1	65
7611	10872	sin keyword	sin location	police investigating ebike collided car little...	1	137
7612	10873	sin keyword	sin location	latest homes razed northern california wildfir...	1	94

7613 rows × 6 columns

In [23]:
```
vectorizer = CountVectorizer(ngram_range=(1,2))  # unigramas y bigramas
X = vectorizer.fit_transform(df["text"])
y = df["target"]
```

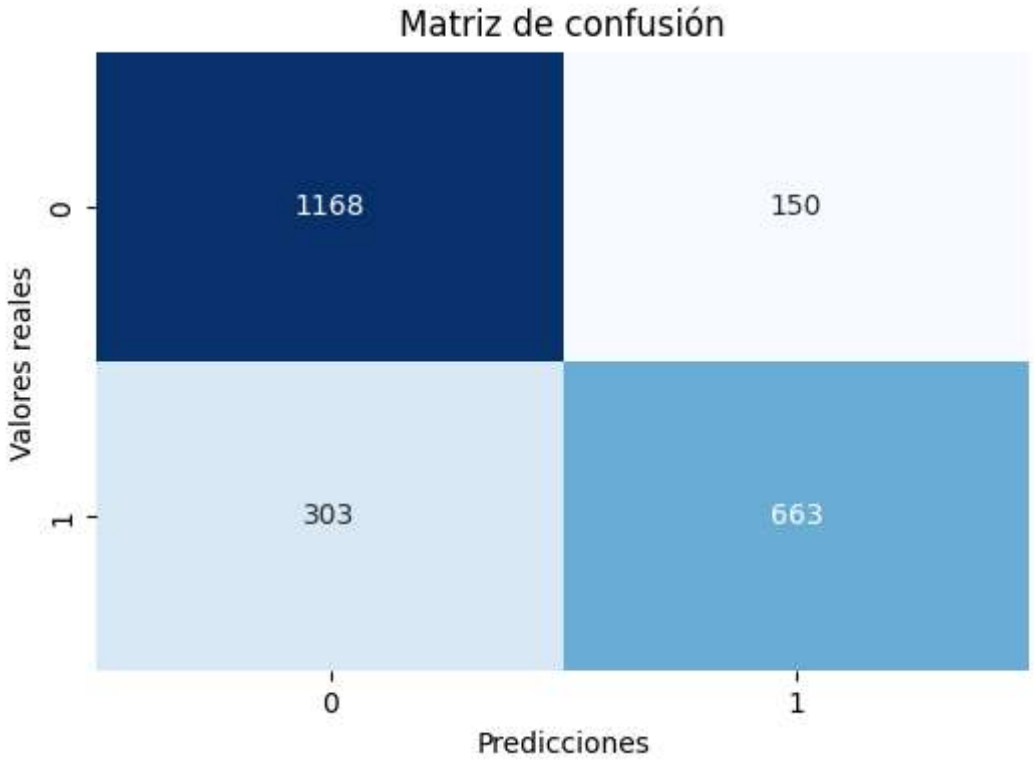
In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Reporte de Clasificación:\n")
print(classification_report(y_test, y_pred))
```

```
Reporte de Clasificación:

              precision    recall  f1-score   support

           0       0.79      0.89      0.84      1318
           1       0.82      0.69      0.75       966

    accuracy                           0.80      2284
   macro avg       0.80      0.79      0.79      2284
weighted avg       0.80      0.80      0.80      2284
```

In [25]:
```python
cm = confusion_matrix(y_test,y_pred)
accuracy=accuracy_score(y_test,y_pred)

print('Matriz de confusión:\n',cm)
print('Accuracy: ',accuracy)
```

```
Matriz de confusión:
 [[1168  150]
 [ 303  663]]
Accuracy:  0.8016637478108581
```

In [26]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, reca

cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # 'weighted' si ha
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Generar matriz
cm = confusion_matrix(y_test, y_pred)

# Crear gráfico
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicciones")
plt.ylabel("Valores reales")
plt.title("Matriz de confusión")
plt.show()

print("Accuracy:", accuracy)
print("Precisión:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
print("\nReporte de clasificación:\n", classification_report(y_test, y_pred))
```




```
Accuracy: 0.8016637478108581
Precisión: 0.8031026766328268
Recall: 0.8016637478108581
F1-Score: 0.7985751564735752

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.79      0.89      0.84      1318
           1       0.82      0.69      0.75       966

    accuracy                           0.80      2284
   macro avg       0.80      0.79      0.79      2284
weighted avg       0.80      0.80      0.80      2284
```