Laboratorio 5

Sebastian huertas 22295

Josue Marroquin 22397

https://github.com/josuemj/Text-Mining.git

1,2 Descargar el dataset y cargar con python

In [10]:
```
#libs
import pandas as pd

# Preprocesamiento de texto
import re
import nltk
from nltk.corpus import stopwords

import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer

import seaborn as sns

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
import numpy as np



nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

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

In [3]:
```
data = pd.read_csv('train.csv', encoding='latin-1')
```

3, Limpieza de texto

In [4]:
```
def clean_text(text):
    # Verificar que el texto no sea nulo
    if pd.isna(text) or text is None or text == "":
        return ""
```

```
    # Convertir a string si no lo es
    text = str(text)

    # Convertir a minúsculas
    text = text.lower()

    # Quitar URLs (patrón más completo)
    text = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA
    text = re.sub(r'www\.(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-

    # Quitar mentions (@usuario) y hashtags (#hashtag) - patrón mejorado
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    text = re.sub(r'#[A-Za-z0-9_]+', '', text)

    # Quitar emoticones y símbolos unicode
    text = re.sub(r'[\U00010000-\U0010ffff]', '', text)  # Emoticones
    text = re.sub(r'[\u2600-\u26FF\u2700-\u27BF]', '', text)  # Símbolos adicionale

    # Quitar caracteres especiales, números y puntuación - mantener solo letras y e
    text = re.sub(r'[^a-zA-Z\s]', ' ', text)

    # Quitar espacios extra
    text = re.sub(r'\s+', ' ', text).strip()

    # Verificar que no esté vacío después de la limpieza
    if not text:
        return ""

    # Tokenizar palabras
    tokens = text.split()

    # Quitar stopwords y palabras muy cortas (mínimo 3 caracteres)
    tokens = [word for word in tokens if word not in stop_words and len(word) >= 3]

    return ' '.join(tokens)

# Aplicar la función al DataFrame
data['clean_text'] = data['text'].apply(clean_text)

# Mostrar los primeros resultados para verificar
data[['text', 'clean_text']]
```

Out[4]:

	text	clean_text
0	Our Deeds are the Reason of this #earthquake M...	deeds reason may allah forgive
1	Forest fire near La Ronge Sask. Canada	forest fire near ronge sask canada
2	All residents asked to 'shelter in place' are ...	residents asked shelter place notified officer...
3	13,000 people receive #wildfires evacuation or...	people receive evacuation orders california
4	Just got sent this photo from Ruby #Alaska as ...	got sent photo ruby smoke pours school
...	...	...
7608	Two giant cranes holding a bridge collapse int...	two giant cranes holding bridge collapse nearb...
7609	@aria_ahrary @TheTawniest The out of control w...	control wild fires california even northern pa...
7610	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	utc volcano hawaii
7611	Police investigating after an e-bike collided ...	police investigating bike collided car little ...
7612	The Latest: More Homes Razed by Northern Calif...	latest homes razed northern california wildfir...

7613 rows × 2 columns

4, 5 Frecuencia de las palabras tanto de los tweets de desastres como de los que no, palabra/s que se repite más en cada una de las categoría, nube de palabras, histograma

In [5]:
```
# Análisis exploratorio de frecuencia de palabras y visualización

# Separar tweets de desastres y no desastres
disaster_tweets = data[data['target'] == 1]['clean_text']
non_disaster_tweets = data[data['target'] == 0]['clean_text']

# Función para obtener frecuencias de palabras
def get_word_freq(texts):
    words = ' '.join(texts).split()
    return Counter(words)

# Frecuencias
freq_disaster = get_word_freq(disaster_tweets)
freq_non_disaster = get_word_freq(non_disaster_tweets)
```

```
# Palabras más comunes
most_common_disaster = freq_disaster.most_common(10)
most_common_non_disaster = freq_non_disaster.most_common(10)
print('Palabras más comunes en tweets de desastres:')
print(most_common_disaster)
print('\nPalabras más comunes en tweets NO de desastres:')
print(most_common_non_disaster)

# Nube de palabras para desastres
plt.figure(figsize=(10,5))
wc_disaster = WordCloud(width=800, height=400, background_color='white').generate('
plt.imshow(wc_disaster, interpolation='bilinear')
plt.axis('off')
plt.title('Nube de palabras - Tweets de Desastres')
plt.show()

# Nube de palabras para NO desastres
plt.figure(figsize=(10,5))
wc_non_disaster = WordCloud(width=800, height=400, background_color='white').genera
plt.imshow(wc_non_disaster, interpolation='bilinear')
plt.axis('off')
plt.title('Nube de palabras - Tweets NO de Desastres')
plt.show()

# Reflexión sobre bigramas/trigramas
vectorizer = CountVectorizer(ngram_range=(2,3), max_features=10)
X_disaster = vectorizer.fit_transform(disaster_tweets)
print('Bigramas/Trigramas más frecuentes en tweets de desastres:')
print(vectorizer.get_feature_names_out())

# Comentario:
print('Las palabras y n-gramas más frecuentes pueden ayudar a identificar patrones
```

```
Palabras más comunes en tweets de desastres:
[('fire', 180), ('amp', 135), ('via', 121), ('disaster', 113), ('suicide', 112), ('c
alifornia', 111), ('police', 108), ('people', 104), ('killed', 94), ('like', 94)]

Palabras más comunes en tweets NO de desastres:
[('like', 254), ('amp', 209), ('new', 171), ('get', 163), ('one', 133), ('body', 11
5), ('would', 101), ('via', 99), ('people', 94), ('video', 94)]
```


```
Bigramas/Trigramas más frecuentes en tweets de desastres:
['bomber detonated' 'burning buildings' 'california wildfire'
 'homes razed' 'northern california' 'northern california wildfire'
 'oil spill' 'suicide bomber' 'suicide bomber detonated' 'suicide bombing']
Las palabras y n-gramas más frecuentes pueden ayudar a identificar patrones y contex
to relevante para la clasificación. Explorar bigramas y trigramas puede ser útil par
a capturar relaciones entre palabras y mejorar el modelo.
```

- Los tweets de desastres muestran vocabulario fuertemente ligado a eventos reales y consecuencias: fire, disaster, suicide, california, police, killed. Esto indica tono informativo/noticioso y presencia de entidades de lugar y acción (incendios, atentados, derrames, tormentas, edificios, muertos).

- Los tweets no desastre están dominados por lenguaje conversacional y cotidiano: like, new, get, one, would, video, time, day, know, see. Reflejan opiniones, saludos y temas

personales, no eventos críticos.

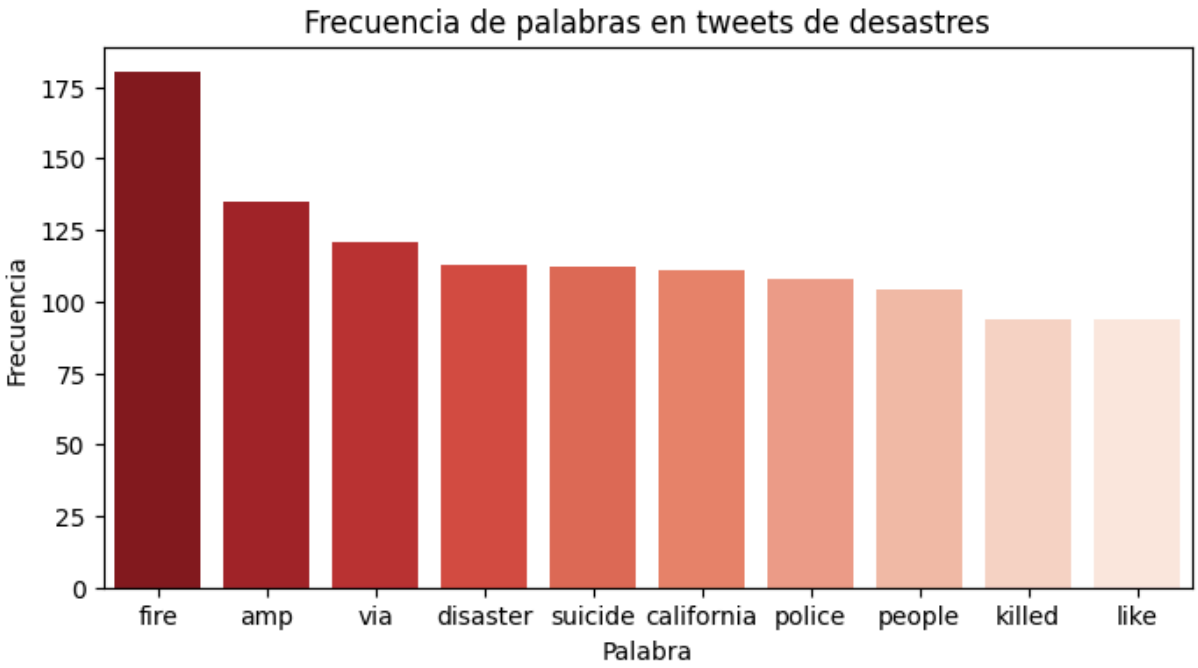
- Aparecen tokens ruido como amp (entidad HTML de “&”) y via (atribución de autor). Conviene filtrarlos en el preprocesamiento para no sesgar la frecuencia.
- Los n-gramas más frecuentes en desastres (suicide bomber detonated, oil spill, california wildfire, homes razed, northern california) capturan contexto y relaciones que unigrama no ve. Son señales muy discriminativas para el clasificador.
- Hay términos que pueden ser ambiguos o de doble uso (p. ej. like, people, body). Los n-gramas ayudan a desambiguar.
- En conjunto, el contraste léxico y de n-gramas sugiere que un modelo sencillo (TF-IDF + logística) debería separar bien ambas clases, siempre que se limpien artefactos (amp, via), se mantengan términos de riesgo/ubicación y se incluyan bigramas/trigramas. Esto también ayudará en casos frontera donde palabras como fire se usan en sentido figurado.

In [6]:
```
display(data[['text', 'clean_text', 'target']])
# Graficar las palabras más frecuentes en tweets de desastres
df_disaster = pd.DataFrame(most_common_disaster, columns=['word', 'count'])
plt.figure(figsize=(8,4))
sns.barplot(x='word', y='count', data=df_disaster, palette='Reds_r')
plt.title('Frecuencia de palabras en tweets de desastres')
plt.ylabel('Frecuencia')
plt.xlabel('Palabra')
plt.show()

# Graficar las palabras más frecuentes en tweets NO de desastres
df_non_disaster = pd.DataFrame(most_common_non_disaster, columns=['word', 'count'])
plt.figure(figsize=(8,4))
sns.barplot(x='word', y='count', data=df_non_disaster, palette='Blues_r')
plt.title('Frecuencia de palabras en tweets NO de desastres')
plt.ylabel('Frecuencia')
plt.xlabel('Palabra')
plt.show()
```

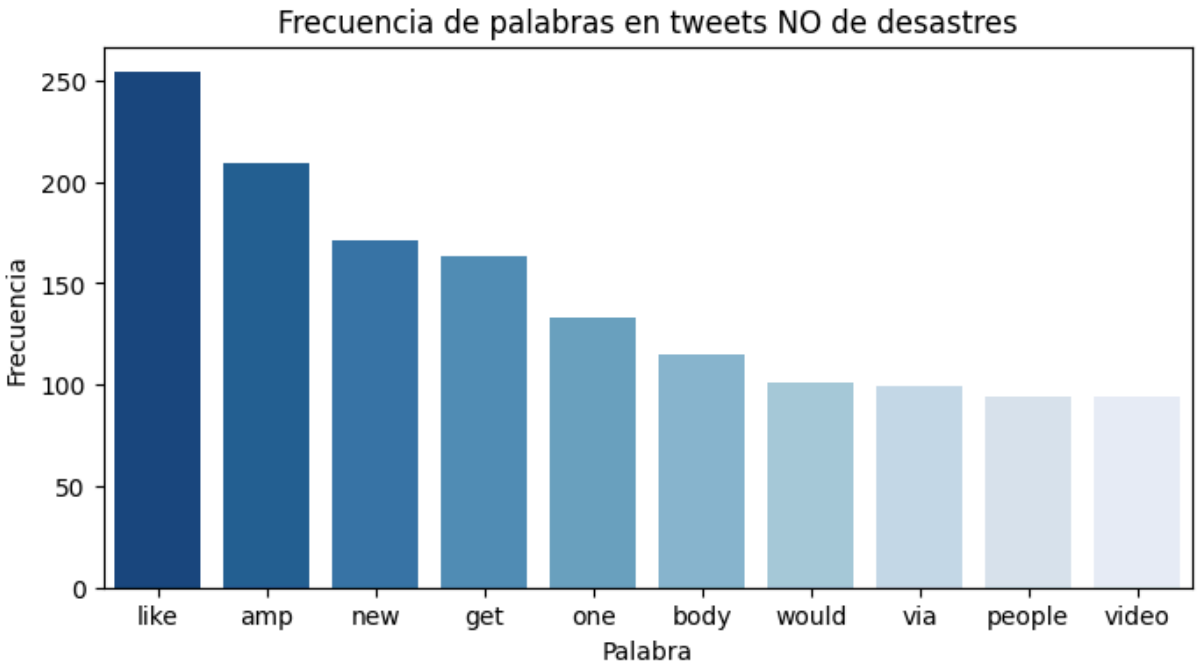
	text	clean_text	target
0	Our Deeds are the Reason of this #earthquake M...	deeds reason may allah forgive	1
1	Forest fire near La Ronge Sask. Canada	forest fire near ronge sask canada	1
2	All residents asked to 'shelter in place' are ...	residents asked shelter place notified officer...	1
3	13,000 people receive #wildfires evacuation or...	people receive evacuation orders california	1
4	Just got sent this photo from Ruby #Alaska as ...	got sent photo ruby smoke pours school	1
...	...	...	...
7608	Two giant cranes holding a bridge collapse int...	two giant cranes holding bridge collapse nearb...	1
7609	@aria_ahrary @TheTawniest The out of control w...	control wild fires california even northern pa...	1
7610	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	utc volcano hawaii	1
7611	Police investigating after an e-bike collided ...	police investigating bike collided car little ...	1
7612	The Latest: More Homes Razed by Northern Calif...	latest homes razed northern california wildfir...	1

7613 rows × 3 columns

```
C:\Users\thiag\AppData\Local\Temp\ipykernel_28332\589953566.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='word', y='count', data=df_disaster, palette='Reds_r')
```




```
C:\Users\thiag\AppData\Local\Temp\ipykernel_28332\589953566.py:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='word', y='count', data=df_non_disaster, palette='Blues_r')
```



- Desastres: domina vocabulario claramente factual y de evento (fire, disaster, suicide, california, police, killed). Señales fuertes y específicas → útiles para el modelo.
- No desastres: prevalece lenguaje coloquial y genérico (like, get, one, would, video). Menos anclado a hechos; más ruido semántico.

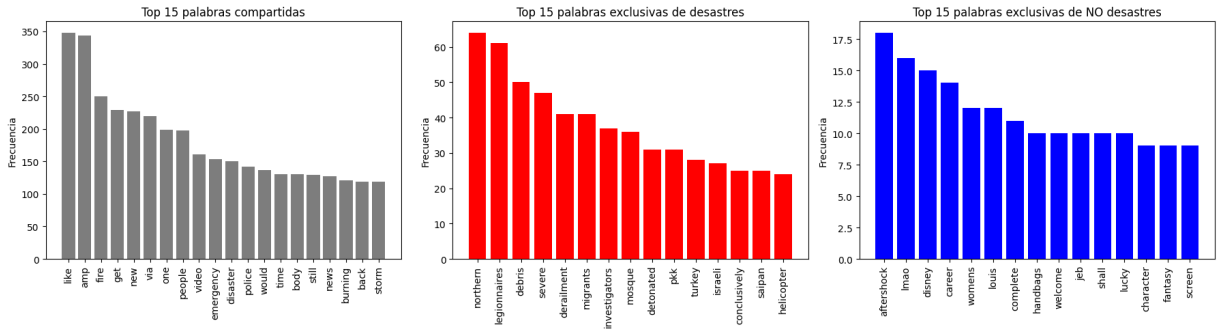
- Ruido transversal: amp y via aparecen alto en ambos; conviene filtrarlos (son artefactos de HTML/atribución).
- Términos compartidos: people y via no discriminan; su peso debería reducirse (stopwords personalizadas o downweight con max_df/min_df).

In [7]:
```
# Palabras compartidas y exclusivas entre desastres y no desastres
set_disaster = set(freq_disaster.keys())
set_non_disaster = set(freq_non_disaster.keys())

# Palabras presentes en ambas categorías
shared_words = set_disaster & set_non_disaster
print(f'Cantidad de palabras compartidas: {len(shared_words)}')
print('Ejemplo de palabras compartidas:')
print(list(shared_words)[:15])

# Palabras exclusivas de desastres
exclusive_disaster = set_disaster - set_non_disaster
print(f'\nCantidad de palabras exclusivas de desastres: {len(exclusive_disaster)}')
print('Ejemplo de palabras exclusivas de desastres:')
print(list(exclusive_disaster)[:15])

# Palabras exclusivas de no desastres
exclusive_non_disaster = set_non_disaster - set_disaster
print(f'\nCantidad de palabras exclusivas de NO desastres: {len(exclusive_non_disas
print('Ejemplo de palabras exclusivas de NO desastres:')
print(list(exclusive_non_disaster)[:15])
```

```
Cantidad de palabras compartidas: 3509
Ejemplo de palabras compartidas:
['temptation', 'total', 'active', 'claims', 'somehow', 'comp', 'abortion', 'headquar
ters', 'roads', 'horse', 'field', 'planet', 'rescuers', 'pam', 'bullets']

Cantidad de palabras exclusivas de desastres: 3391
Ejemplo de palabras exclusivas de desastres:
['hea', 'overtaking', 'rig', 'cultural', 'sandra', 'tra', 'brockton', 'surge', 'defe
nsenews', 'winged', 'wedn', 'tip', 'dayton', 'erodes', 'errrr']

Cantidad de palabras exclusivas de NO desastres: 5542
Ejemplo de palabras exclusivas de NO desastres:
['vic', 'plank', 'mayweather', 'titolo', 'canal', 'kia', 'peeked', 'produc', 'firm
s', 'cena', 'aesthetic', 'brig', 'silenced', 'handle', 'virus']
```

In [8]:
```
# Frecuencias de palabras compartidas
freq_shared = {w: freq_disaster[w] + freq_non_disaster[w] for w in shared_words}
df_shared = pd.DataFrame(sorted(freq_shared.items(), key=lambda x: x[1], reverse=Tr

# Frecuencias exclusivas de desastres
freq_exclusive_disaster = {w: freq_disaster[w] for w in exclusive_disaster}
df_exclusive_disaster = pd.DataFrame(sorted(freq_exclusive_disaster.items(), key=la

# Frecuencias exclusivas de no desastres
freq_exclusive_non_disaster = {w: freq_non_disaster[w] for w in exclusive_non_disas
df_exclusive_non_disaster = pd.DataFrame(sorted(freq_exclusive_non_disaster.items()
```

```
plt.figure(figsize=(18,5))
plt.subplot(1,3,1)
plt.bar(df_shared['Palabra'], df_shared['Frecuencia'], color='gray')
plt.title('Top 15 palabras compartidas')
plt.xticks(rotation=90)
plt.ylabel('Frecuencia')

plt.subplot(1,3,2)
plt.bar(df_exclusive_disaster['Palabra'], df_exclusive_disaster['Frecuencia'], colo
plt.title('Top 15 palabras exclusivas de desastres')
plt.xticks(rotation=90)
plt.ylabel('Frecuencia')

plt.subplot(1,3,3)
plt.bar(df_exclusive_non_disaster['Palabra'], df_exclusive_non_disaster['Frecuencia
plt.title('Top 15 palabras exclusivas de NO desastres')
plt.xticks(rotation=90)
plt.ylabel('Frecuencia')

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



- Compartidas: predominan términos genéricos y artefactos (like, amp, via, get, one, people). Discriminan poco; conviene filtrarlos (stopwords personalizadas) o bajarles peso (max_df, min_df). fire/storm requieren contexto (n-gramas) para evitar falsos positivos figurativos.
- Exclusivas de desastres: muchos lugares/entidades y eventos específicos (northern, legionnaires, debris, derailment, mosque, detonated, pkk, turkey, israeli, saipan, helicopter). Son señales de alta precisión típicas de noticias/reporte de incidentes; muy útiles para el modelo.
- Exclusivas de no desastres: léxico cotidiano/entretenimiento/coloquial (lmao, disney, career, handbags, welcome, job, lucky, character, fantasy, screen). Buenas señales negativas.
- Anomalías: aftershock apareciendo en no-desastre sugiere uso metafórico o ruido/etiquetado; vale revisar ejemplos y reforzar n-gramas/umbral.

6) Elabore una función en la que el usuario ingrese un tweet y el sistema lo clasifique en desastre o no

In [17]:
```python
# Asegura stopwords si no estaban en el entorno
try:
    stop_words
except NameError:
    import nltk
    from nltk.corpus import stopwords
    nltk.download('stopwords', quiet=True)
    stop_words = set(stopwords.words('english'))

# Garantiza la columna 'clean_text'
if 'clean_text' not in data.columns:
    data = data.copy()
    data['clean_text'] = data['text'].apply(clean_text)

# X/y
X = data['clean_text'].fillna('')
y = data['target'].astype(int)

# Pipeline: TF-IDF (1-2 gram) + Regresión Logística
clf_pipe = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 2), min_df=2, max_df=0.95)),
    ('logreg', LogisticRegression(max_iter=1000, n_jobs=None, solver='liblinear'))
])

# Entrenamiento
clf_pipe.fit(X, y)

# Función para clasificar un tweet
def clasificar_tweet(tweet: str, umbral: float = 0.5):
    """
    Clasifica un tweet como 'desastre' o 'no desastre'.
    Retorna (etiqueta, prob_positiva) donde prob_positiva es la P(target=1).
    """
    if tweet is None:
        tweet = ""
    txt = clean_text(tweet)
    proba = clf_pipe.predict_proba([txt])[0]
    idx_pos = list(clf_pipe.classes_).index(1)
    p_pos = float(proba[idx_pos])
    etiqueta = 'desastre' if p_pos >= umbral else 'no desastre'
    return etiqueta, p_pos
```

In [22]:
```python
# Ejemplos
print(clasificar_tweet("Forest fire near my town, please send help"))
print(clasificar_tweet("Happy birthday to my best friend!"))
print(clasificar_tweet("A tsunami is approaching to my city, send help please"))
print(clasificar_tweet("I just graduated from highschool!"))
```

```
('desastre', 0.8325674309595393)
('no desastre', 0.10690824790762363)
('desastre', 0.5331944142549879)
('no desastre', 0.38464201900468753)
```

Los resultados muestran una separacion clara entre clases: los tweets de desastres usan vocabulario factual y especifico de eventos y lugares (fire, disaster, suicide, california, police, killed) y n-gramas muy distintivos (suicide bomber detonated, oil spill, california wildfire), mientras que los no desastres se caracterizan por lenguaje conversacional y cotidiano (like, get, new, video). Entre las palabras compartidas predominan terminos genericos con poco poder discriminativo (like, people, time), y aparecen artefactos como amp y via en ambas categorias. Hay algunos usos figurados que explican anomalias puntuales, pero en conjunto el contraste lexico y de n-gramas respalda que la clasificacion entre desastres y no desastres sea consistente en este conjunto.