

Universidad del Valle de Guatemala
Data Science - CC3066 - Sección 20
Catedrático: LUIS ROBERTO FURLAN COLLVER

Integrantes:

JULIO ROBERTO HERRERA SABAN

OSCAR RENE SARAVIA DONIS

DIEGO DE JESUS ARREDONDO TURCIOS

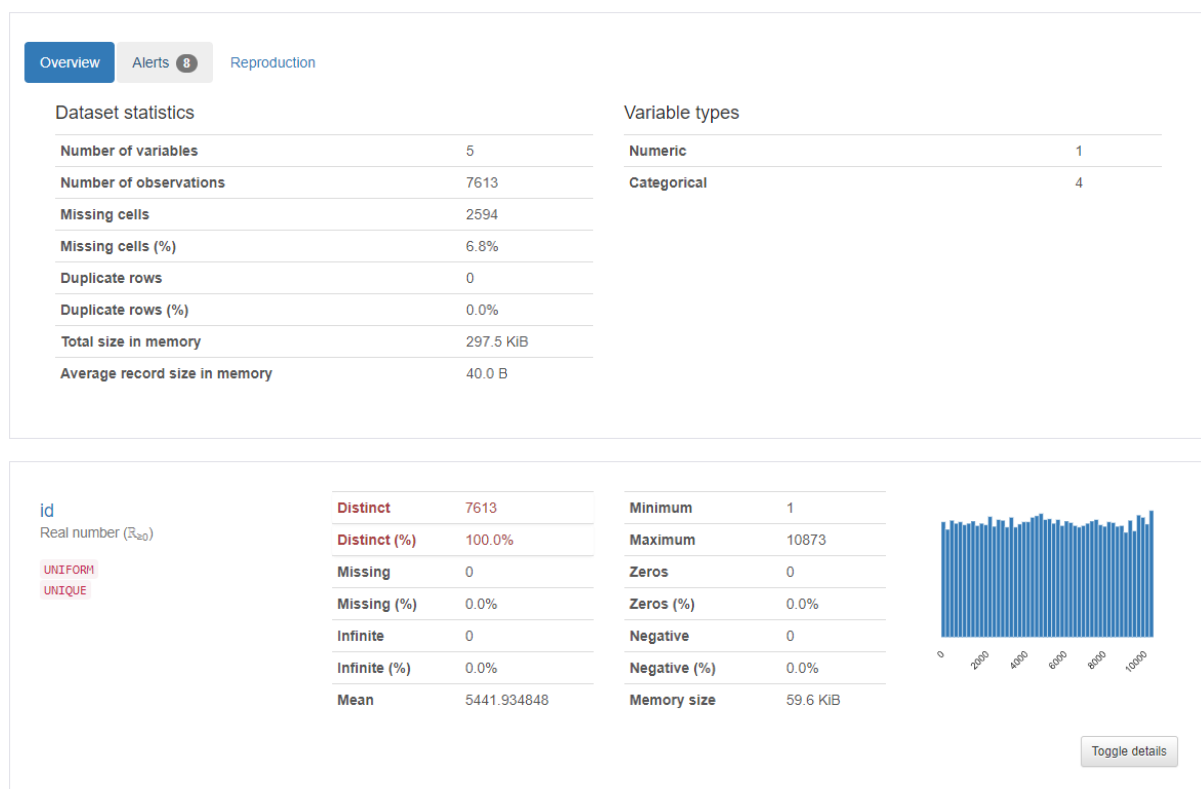
Laboratorio 5 Análisis de Sentimientos

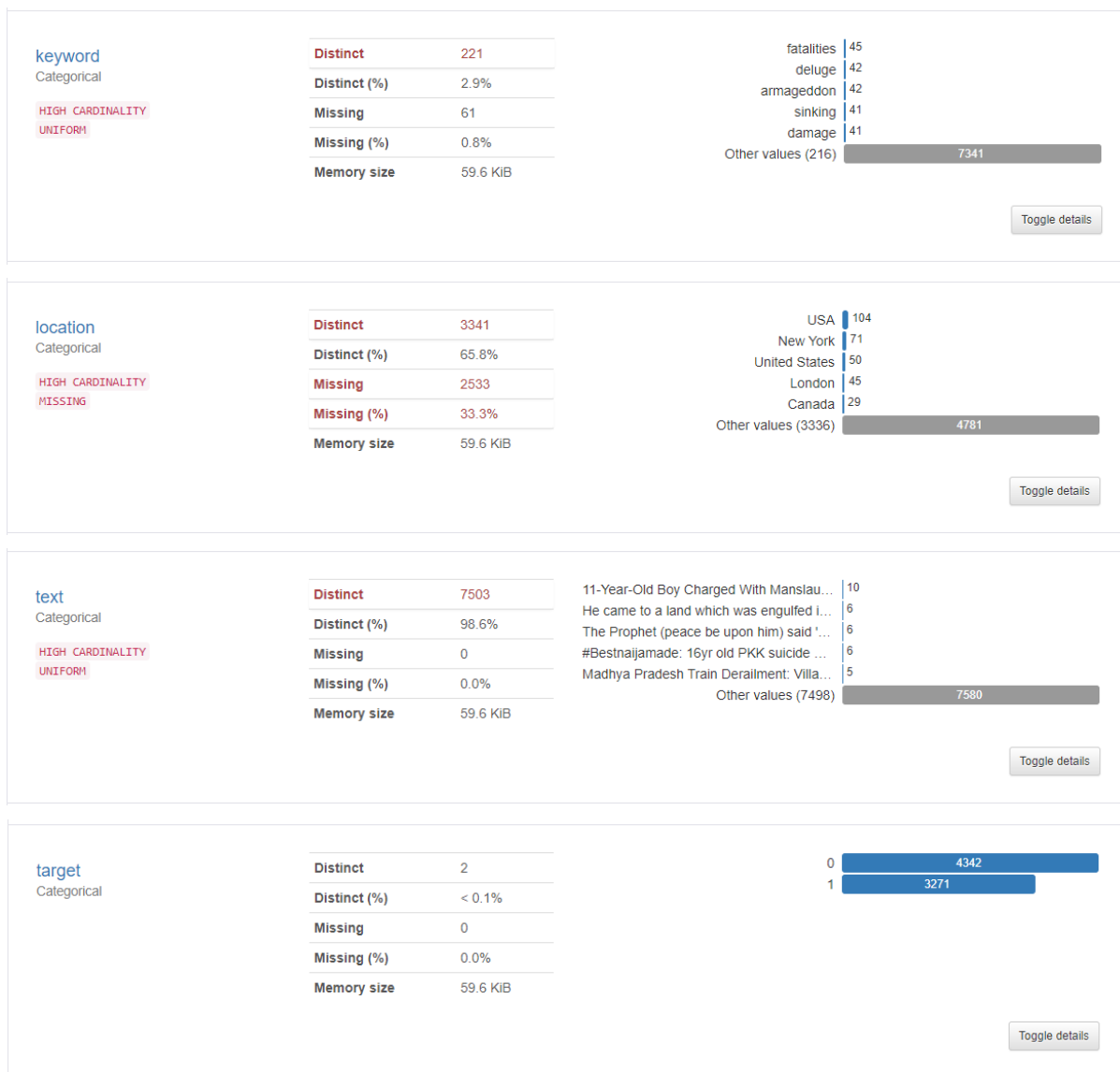
Link del Repositorio: <https://github.com/arr19422/Lab5-DS>

(15 puntos) Análisis exploratorio

El *dataset* utilizado fue adquirido de Kaggle: [Natural Language Processing with Disaster Tweets](#) y se generó un reporte al *dataset* sin procesamiento.

Reporte:





Las variables del dataset representan lo siguiente:

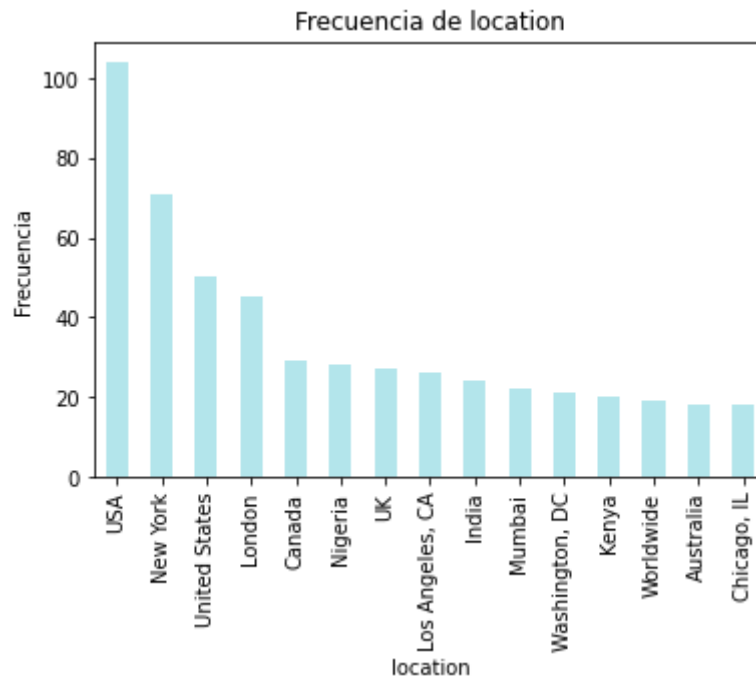
- id: Identificador del tweet.
- keyword: Palabra clave del tweet.
- location: Ubicación desde donde se envió el tweet.
- text: Cuerpo del tweet.
- target: Variable boolean que es verdadero si el tweet habla de un desastre real y falso si no.

Como podemos ver, las variables *keyword* y *location* tienen 0.8% y 33.3% de valores vacíos sin embargo, no se removerán estas observaciones ya que si bien podemos obtener conclusiones de ellas, es la variable *text* la que nos importa para realizar la clasificación.

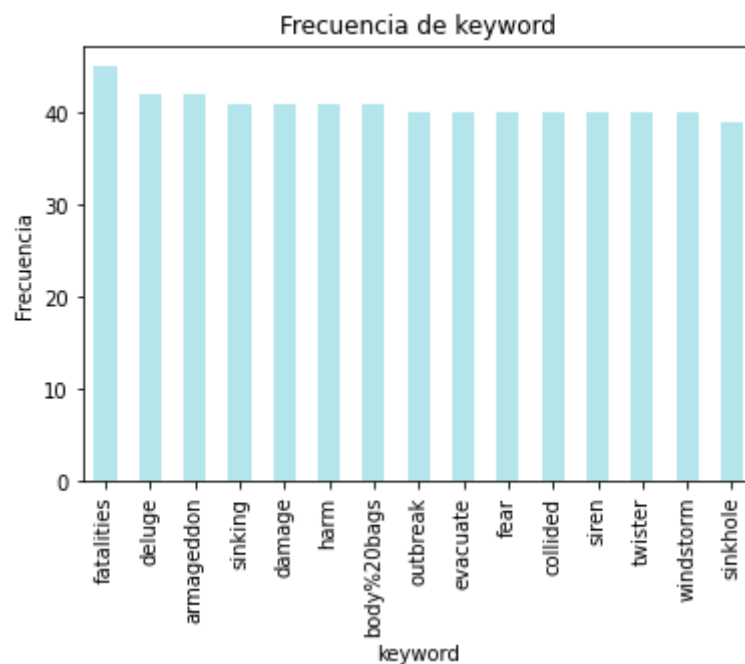
Hallazgos:

Algunas hallazgos encontrados por medio de gráficas a las variables son:

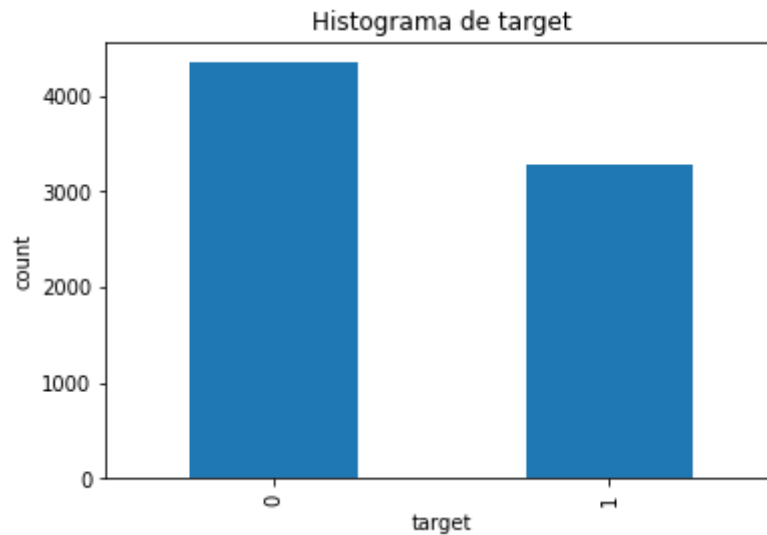
La ubicación desde donde más se tuiteó fue USA, sin embargo hay distintos valores que representan a este mismo país ya que también se encuentra como United States o como todos los estados específicos del país como New York, Los Angeles, CA y Washington, DC que se encuentran también entre el top 11 de ubicaciones más repetidas.



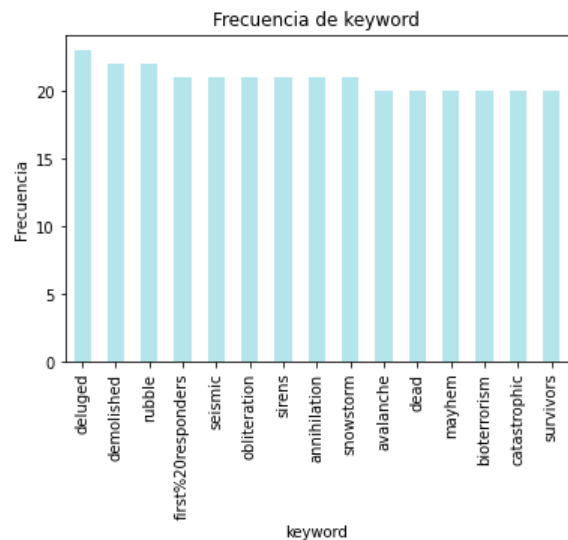
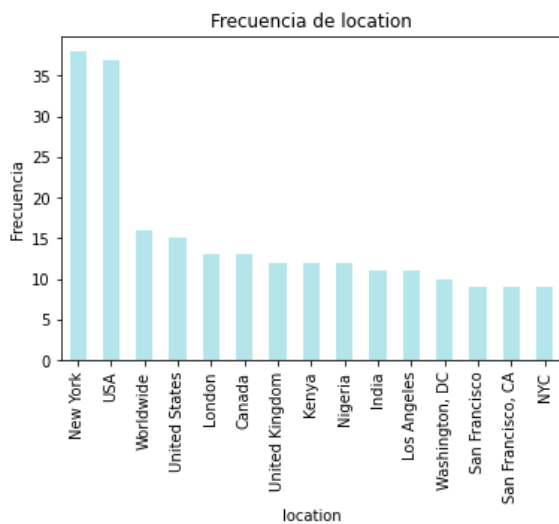
Las palabras clave (*keywords*) tienen una distribución más uniforme, repitiéndose más de la mitad de las observaciones entre 30 y 45 veces. Se puede observar que entre el top 15 de palabras más frecuentes como *keywords* están relacionadas con algún hecho trágico. También se observa que algunos valores pueden estar conformados realmente por más de una palabra pero siendo el espacio representado por "%20" normalmente utilizado para hacer *encoding* de URLs.



La variable *target* es la que nos ayudará a entrenar el modelo y clasificar si se habla de un desastre real o no. Esta es la variable que no se encuentra en el *dataset* de *testeo* y en este de entrenamiento muestra que existe mayoría de tweets que no hablan de un desastre real, sin embargo es una distribución bastante buena.



En cuanto al *dataset* de *testeo* se observó un comportamiento parecido en las variables *location* y *keywords* con algunas diferencias en las posiciones de los valores con más frecuencia.



(15 puntos) Limpieza y procesamiento de los datos

Primero se convirtieron todos los datos a minúsculas:

Convirtiendo a minúsculas

```
for i in range(len(train)):
    train.loc[i, 'text'] = train.loc[i, 'text'].lower()

for i in range(len(test)):
    test.loc[i, 'text'] = test.loc[i, 'text'].lower()
```

✓ 2.1s

```
train.head()
```

✓ 0.6s

	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 longe 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

```
test.head()
```

✓ 0.4s

	id	keyword	location	text
0	0	NaN	NaN	just happened a terrible car crash
1	2	NaN	NaN	heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	apocalypse lighting. #spokane #wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 in china and taiwan

Luego se quitaron los URLs:

Quitando urls

```
#Quitar las urls
for i in range(len(train)):
    train.loc[i, 'text'] = re.sub(r'((http|https)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-\_=#]+\.[a-zA-Z]{2,6}([a-zA-Z0-9\.\&\/\?\:@\-\_=#])*', '', train.loc[i, 'text'])

for i in range(len(test)):
    test.loc[i, 'text'] = re.sub(r'((http|https)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-\_=#]+\.[a-zA-Z]{2,6}([a-zA-Z0-9\.\&\/\?\:@\-\_=#])*', '', test.loc[i, 'text'])
```

✓ 2.2s

```
train.head()
```

✓ 0.5s

	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

```
test.head()
```

✓ 0.4s

	id	keyword	location	text
0	0	NaN	NaN	just happened a terrible car crash
1	2	NaN	NaN	heard about #earthquake is different cities. s...
2	3	NaN	NaN	there is a forest fire at spot pond. geese are...
3	9	NaN	NaN	apocalypse lighting. #spokane #wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 in china and taiwan

Seguidamente se quitaron signos de puntuación y los caracteres especiales:

Eliminando signos de puntuación y caracteres especiales (@, #, \$, %, etc.)

```
#Se eliminan los signos de puntuacion
for i in range(len(train)):
    train.loc[i, 'text'] = train.loc[i, 'text'].translate(str.maketrans(string.punctuation, ' ' * len(string.punctuation)))

for i in range(len(test)):
    test.loc[i, 'text'] = test.loc[i, 'text'].translate(str.maketrans(string.punctuation, ' ' * len(string.punctuation)))
```

✓ 2.2s

```
train.head()
```

✓ 0.4s

	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

```
test.head()
```

✓ 0.6s

	id	keyword	location	text
0	0	NaN	NaN	just happened a terrible car crash
1	2	NaN	NaN	heard about earthquake is different cities s...
2	3	NaN	NaN	there is a forest fire at spot pond geese are...
3	9	NaN	NaN	apocalypse lighting spokane wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 in china and taiwan

Luego se quitaron los stopwords, definiendo el idioma inglés.

Definiendo stopwords en idioma inglés Quitando las stopwords (the, a, an, etc.)

```
#Ref: https://www.delftstack.com/howto/python/remove-stop-words-in-python/#:~:text=The%20nltk%20(Natural%20Language%20Processing,the%20list%20from%20this%20library.
#Quitar las stopwords
stopWords = stopwords.words('english')

for i in range(len(train)):
    train.loc[i,'text'] = ' '.join(word for word in train.loc[i,'text'].split() if word not in stopWords)

for i in range(len(test)):
    test.loc[i,'text'] = ' '.join(word for word in test.loc[i,'text'].split() if word not in stopWords)
```

```
train.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	residents asked shelter place notified officer...	1
3	6	NaN	NaN	13 000 people receive wildfires evacuation ord...	1
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou...	1

```
test.head()
```

	id	keyword	location	text
0	0	NaN	NaN	happened terrible car crash
1	2	NaN	NaN	heard earthquake different cities stay safe ev...
2	3	NaN	NaN	forest fire spot pond geese fleeing across str...
3	9	NaN	NaN	apocalypse lighting spokane wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 china taiwan

Posteriormente se quitan los emojis:

```
train.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	residents asked shelter place notified officer...	1
3	6	NaN	NaN	13 000 people receive wildfires evacuation ord...	1
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou...	1

```
test.head()
```

	id	keyword	location	text
0	0	NaN	NaN	happened terrible car crash
1	2	NaN	NaN	heard earthquake different cities stay safe ev...
2	3	NaN	NaN	forest fire spot pond geese fleeing across str...
3	9	NaN	NaN	apocalypse lighting spokane wildfires
4	11	NaN	NaN	typhoon soudelor kills 28 china taiwan

Y por último, se quitan los números, pero se decidió dejar el 911 por futura utilidad al momento de analizar los datos

Quitando números excepto el 911

```
#Quitar numeros, se considera dejar el 911 ya que es un numero representativo en EEUU y puede ser de utilidad
for i in range(len(train)):
    train.loc[i,'text'] = re.sub(r'#\S+|\d+', lambda match: match.group(0) if match.group(0).startswith('911') else '', train.loc[i,'text'])

for i in range(len(test)):
    test.loc[i,'text'] = re.sub(r'#\S+|\d+', lambda match: match.group(0) if match.group(0).startswith('911') else '', test.loc[i,'text'])
```

```
train.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	residents asked shelter place notified officer...	1
3	6	NaN	NaN	people receive wildfires evacuation orders c...	1
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou...	1

```
test.head()
```

	id	keyword	location	text
0	0	NaN	NaN	happened terrible car crash
1	2	NaN	NaN	heard earthquake different cities stay safe ev...
2	3	NaN	NaN	forest fire spot pond geese fleeing across str...
3	9	NaN	NaN	apocalypse lighting spokane wildfires
4	11	NaN	NaN	typhoon soudelor kills china taiwan

Durante todo este proceso pudimos ir viendo cómo cambiaban - al menos - las primeras observaciones en su columna *text*, por lo que comprobamos que los valores en ambos datasets fueron arreglados según lo indicado.

(20 puntos) Clasificación de Palabras

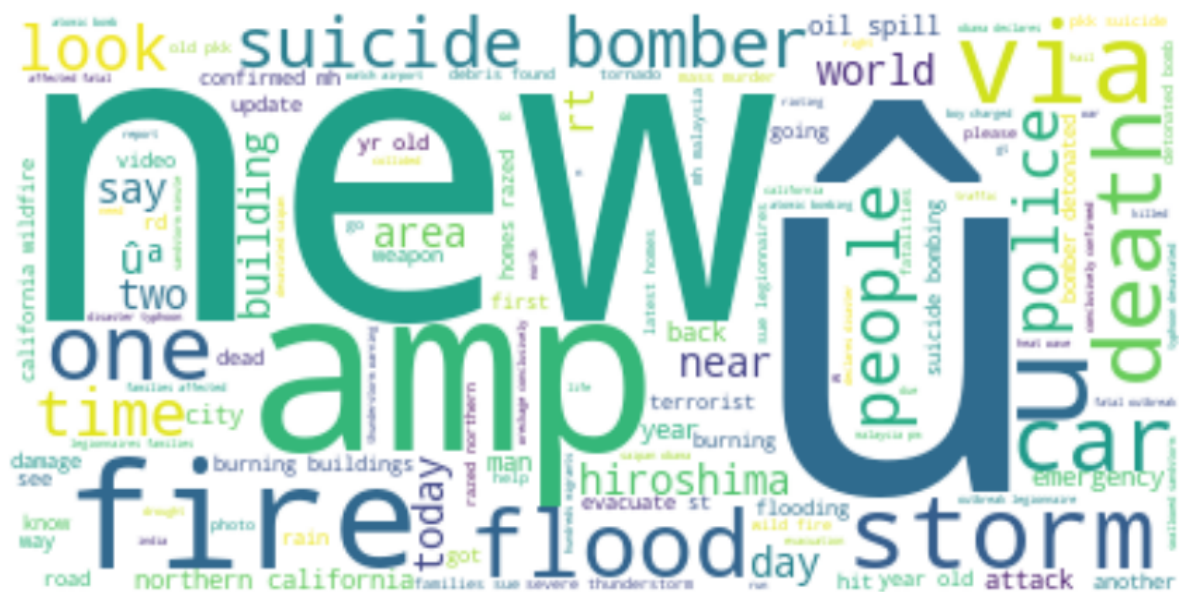
Primero separamos los tweets que son de desastres reales y los que no a partir del valor de la variable *target*.

Frecuencia de tweets de desastres

```
disaster_freq = train[train['target'] == 1]
disaster_words = []
for i in range(len(disaster_freq)):
    text = disaster_freq.iloc[i]['text'].split()
    for word in text:
        disaster_words.append(word)

keywords = " ".join(review for review in disaster_words)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(keywords)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

1 ✓ 0.6s



✓ 0.8s



Analizando bigramas

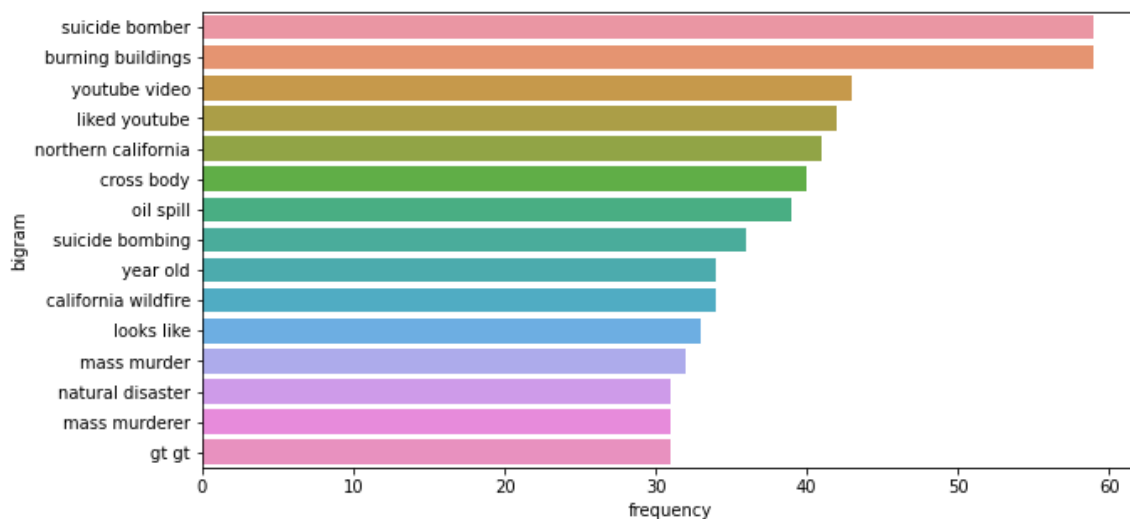
[+ Code](#)

```
# https://towardsdatascience.com/text-analysis-basics-in-python-443282942ec5
def get_bigrams(text):
    count_vectorizer = CountVectorizer(ngram_range=(2, 2))
    ngrams = count_vectorizer.fit_transform(text)
    count_values = ngrams.toarray().sum(axis=0)
    vocab = count_vectorizer.vocabulary_
    df_ngram = pd.DataFrame(sorted([(count_values[i], k) for k, i in vocab.items()],
                                   reverse=True)).rename(columns={0: 'frequency', 1: 'bigram'})
    return df_ngram

plt.figure(figsize=(10, 5))
bigrams = get_bigrams(train['text'])
x = bigrams['frequency'][:15]
y = bigrams['bigram'][:15]
sns.barplot(x=x, y=y)
```

✓ 1.3s

<AxesSubplot:xlabel='frequency', ylabel='bigram'>



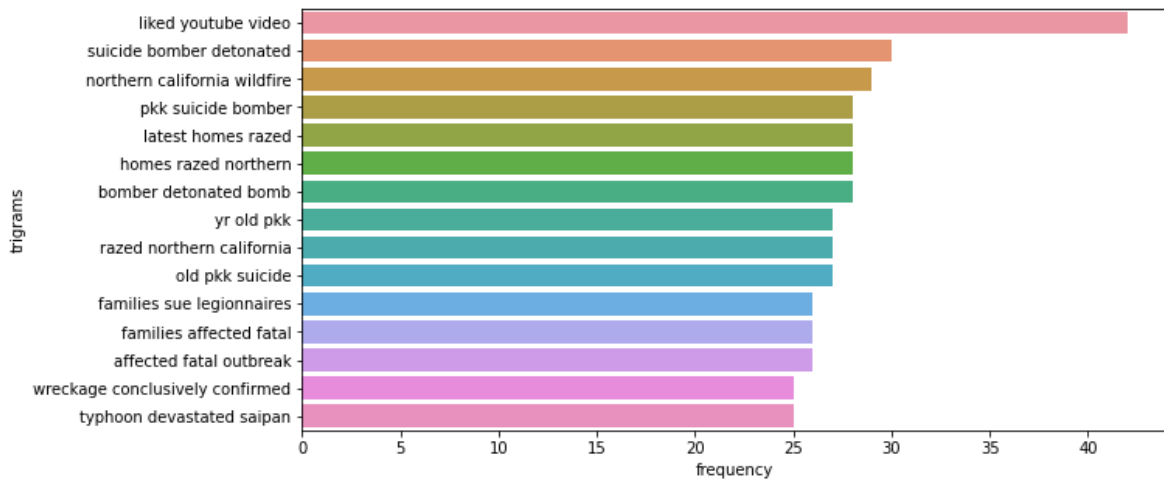
Analizando trigramas

```
# https://towardsdatascience.com/text-analysis-basics-in-python-443282942ec5
def get_trigrams(text):
    count_vectorizer = CountVectorizer(ngram_range=(3, 3))
    ngrams = count_vectorizer.fit_transform(text)
    count_values = ngrams.toarray().sum(axis=0)
    vocab = count_vectorizer.vocabulary_
    df_ngram = pd.DataFrame(sorted([(count_values[i], k) for k, i in vocab.items()]), reverse=True).rename(columns={0: 'frequency', 1: 'trigrams'})
    return df_ngram

plt.figure(figsize=(10, 5))
trigrams = get_trigrams(train['text'])
x = trigrams['frequency'][:15]
y = trigrams['trigrams'][:15]
sns.barplot(x=x, y=y)
```

✓ 1.3s

<AxesSubplot:xlabel='frequency', ylabel='trigrams'>



(10 puntos) Clasificación de tweets en positivo, negativo y neutro.

```
train.head()
```

✓ 0.9s

	id	keyword	location	text	target	score	sentiment
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	1	0.2732	neutral
1	4	NaN	NaN	forest fire near la ronge sask canada	1	-0.3400	neutral
2	5	NaN	NaN	residents asked shelter place notified officer...	1	0.0000	neutral
3	6	NaN	NaN	people receive wildfires evacuation orders c...	1	0.0000	neutral
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou...	1	0.0000	neutral

```
test.head()
```

✓ 0.4s

	id	keyword	location	text	score	sentiment
0	0	NaN	NaN	happened terrible car crash	-0.7003	negative
1	2	NaN	NaN	heard earthquake different cities stay safe ev...	0.4404	neutral
2	3	NaN	NaN	forest fire spot pond geese fleeing across str...	-0.6159	negative
3	9	NaN	NaN	apocalypse lighting spokane wildfires	0.0000	neutral
4	11	NaN	NaN	typhoon soudelor kills china taiwan	-0.5423	negative

Referencias

Akladyous, Boula. (2021). entiment Analysis using VADER. Medium. Extraído de: <https://akladyous.medium.com/sentiment-analysis-using-vader-c56bcffe6f24>

Yang, Sophia. (2020). Text analysis basics in Python. Towards Data Science. Extraído de: <https://towardsdatascience.com/text-analysis-basics-in-python-443282942ec5>