

### Avances lab #4

#### Parte 3. Limpiar y preprocesar los datos

##### 1. Carga y Limpieza de Datos

El código comienza cargando un archivo CSV llamado 'train\_limpios.csv' y realiza una limpieza básica reemplazando los valores vacíos con NA (Not Available).

```
```R  
data<-read.csv('train_limpios.csv')  
data[data == ""]<-NA  
...``
```

		id	keyword	location	\
0		0	nan	nan	
1		2	nan	nan	
2		3	nan	nan	
3		9	nan	nan	
4		11	nan	nan	
...		...	...	...	
3258		10861	nan	nan	
3259		10865	nan	nan	
3260		10868	nan	nan	
3261		10874	nan	nan	
3262		10875	nan	nan	

```
text
0      just happened a terrible car crash
1  heard about #earthquake is different cities, s...
2  there is a forest fire at spot pond, geese are...
3      apocalypse lighting. #spokane #wildfires
4      typhoon soudelor kills 28 in china and taiwan
...
3258  earthquake safety los angeles  ûò safety faste...
3259  storm in ri worse than last hurricane. my city...
3260  green line derailment in chicago http://t.co/u...
3261  meg issues hazardous weather outlook (hwo) htt...
3262  #cityofcalgary has activated its municipal eme...
```

#### Estadísticas de longitud de textos:

```
count    7613.000000
mean     8.533824
std      3.208316
min     3.000000
25%     6.000000
50%     8.000000
75%    10.000000
max    20.000000
Name: tweet_length, dtype: float64
```

Cantidad de categorías únicas en la ubicación: 3129

#### Estadísticas de longitud de textos:

```
count    7613.000000
mean     93.899251
std      31.709694
min     6.000000
25%    71.000000
50%    99.000000
75%   123.000000
max   143.000000
Name: tweet_length, dtype: float64
```

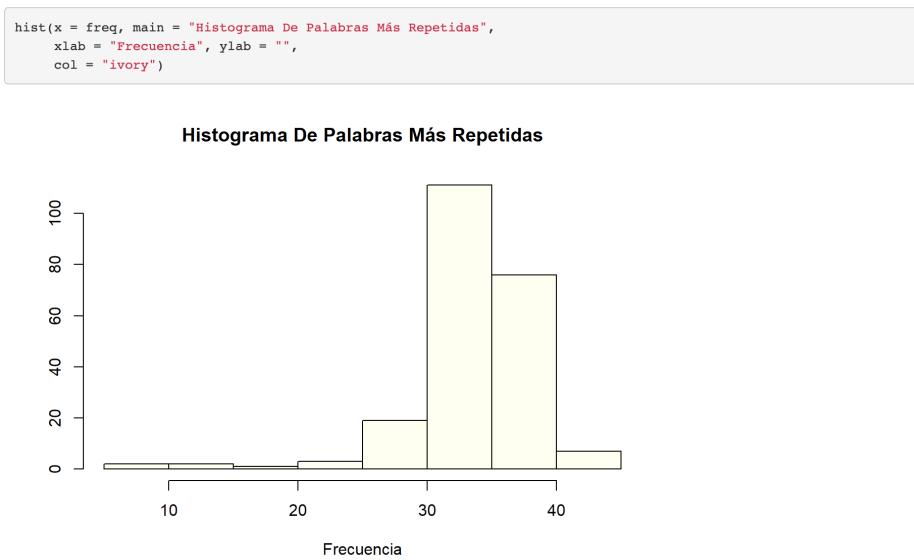
Cantidad de categorías únicas en base al texto: 3129

Parte 4. Obtenga la frecuencia de las palabras tanto de los tweets de desastres como de los que no. ¿Qué palabras cree que le servirán para hacer un mejor modelo de clasificación? ¿Vale la pena explorar bigramas o trigramas para analizar el contexto?

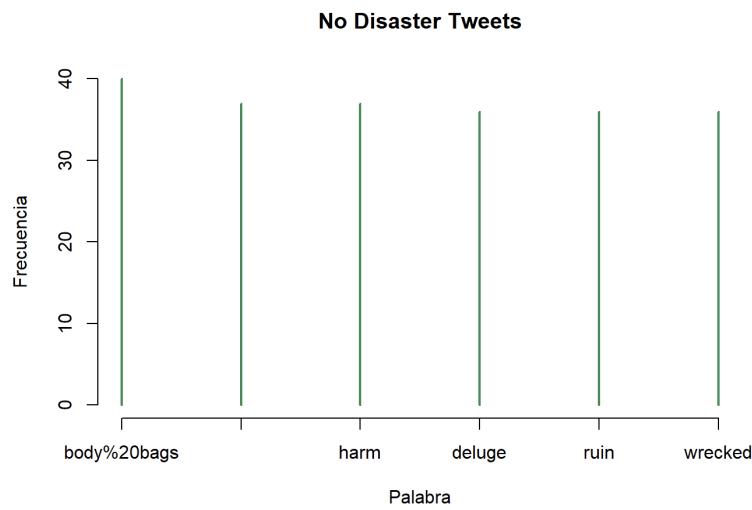
## Frecuencia de palabras



También se realizó un histograma de las palabras más repetidas para ver como es la distribución de la misma.



Ahora un gráfico de las palabras que más se repiten en la categoría de "No Disaster".

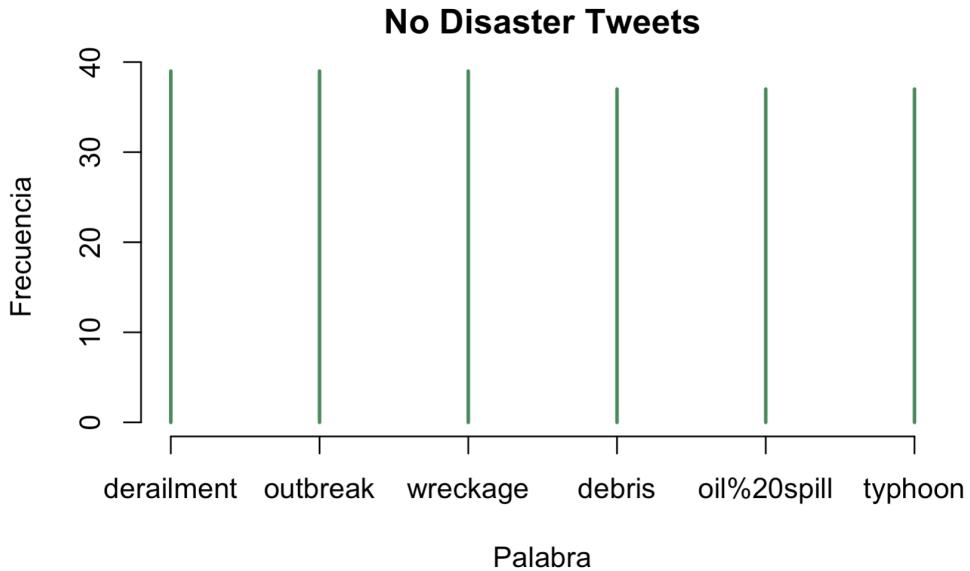


Vemos que la palabra que más se repite en la categoría de tweets que no tratan de desastres es "body bags".

También se realizó una nube de palabras de esta misma categoría para ver la frecuencia de cada una.

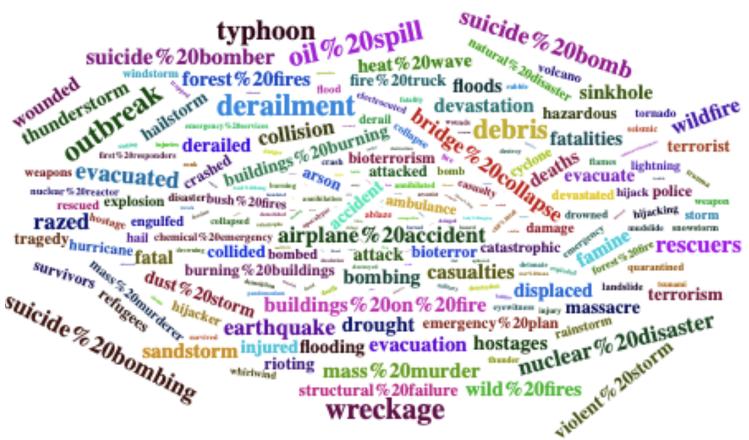


Ahora vamos con las palabras que más se repiten en la categoría de "Disaster", que igual están representadas en un gráfico:



Se observa que la palabra que más se repite en la categoría de tweets que tratan sobre desastres es "derailment".

También se realizó una nube de palabras de esta misma categoría para ver la frecuencia de cada una.

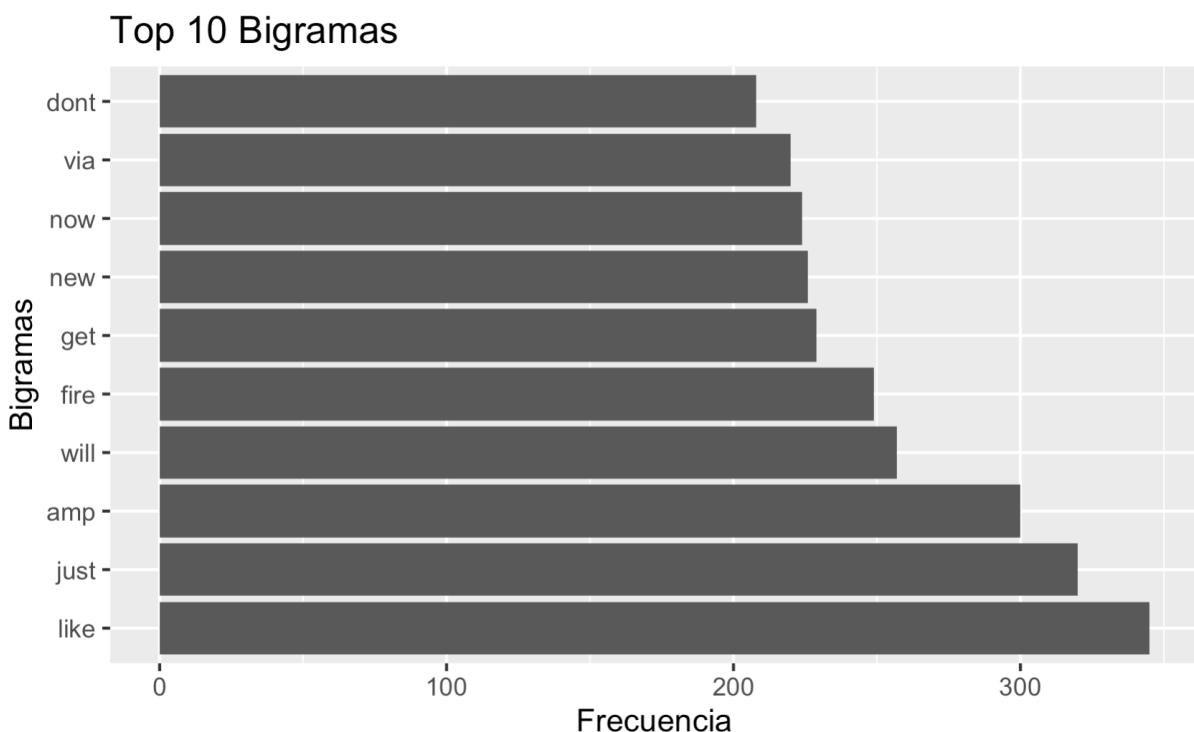


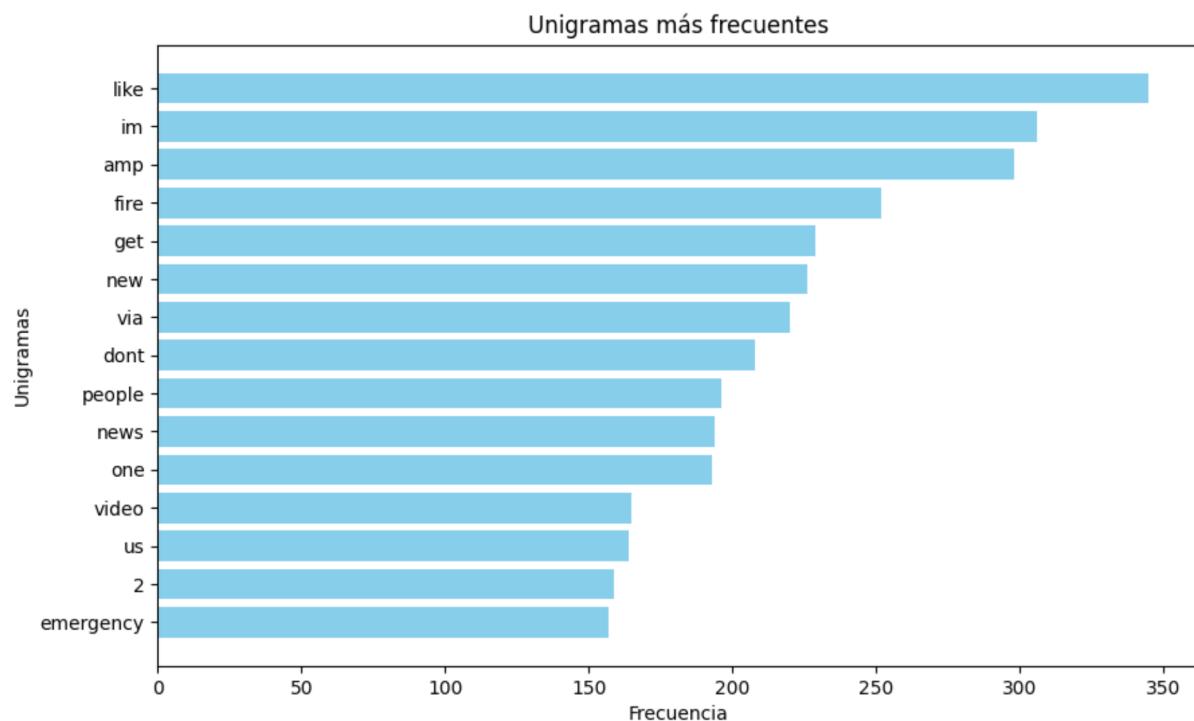
Por lo menos en las primeras seis palabras de cada categoría no se repite ni una palabra, es decir no hay palabras que estén en ambas categorías.

Bigramas construidos desde los tweets, calculado sus frecuencias y también sus probabilidades. El resultado que se muestra son los bigramas más comunes junto con sus respectivas frecuencias y probabilidades.

	<b>bigram</b> <code>&lt;chr&gt;</code>	<b>frequency</b> <code>&lt;dbl&gt;</code>	<b>probability</b> <code>&lt;dbl&gt;</code>
the	the	3269	0.040572407
and	and	1420	0.017623988
for	for	893	0.011083255
you	you	797	0.009891774
with	with	572	0.007099240
that	that	562	0.006975128

6 rows





## 2. Análisis de Frecuencias

El siguiente segmento de código crea una tabla de frecuencias de la columna 'keyword'. Esta tabla nos da una idea de cuán frecuentemente aparece cada palabra clave.

```
```R
freq<-table(data$keyword, useNA = 'no')
```

```

## 3. Cruce de Variables con la columna 'target'

Se realiza un cruce de datos entre 'keyword' y 'target'. Aquí estamos categorizando los datos en dos conjuntos basados en la columna 'target': tweets relacionados con desastres ('target == 1') y tweets no relacionados con desastres ('target == 0').

```
```R
no_disaster<-subset(x = data, subset = target == 0, select = c("keyword"))
freq_no_disaster<-table(no_disaster$keyword)

disaster<-subset(x = data, subset = target == 1, select = c("keyword"))
freq_disaster<-table(disaster$keyword)
```

```

## 4. Visualización de Datos

### 1. Nube de Palabras:

- Una nube de palabras se genera para toda la columna "keyword" para visualizar la frecuencia de cada palabra clave.
- También se generan nubes de palabras específicas para las categorías "no desastre" y "desastre".

```
```R
w1<-wordcloud2(data = freq, size = 0.1, shape = "cloud", color="random-dark", ellipticity =
0.5)
w2<-wordcloud2(data = freq_no_disaster, size = 0.1, shape = "cloud", color="random-dark",
ellipticity = 0.5)
w3<-wordcloud2(data = freq_disaster, size = 0.1, shape = "cloud", color="random-dark",
ellipticity = 0.5)
```

```

## 2. Histograma de Frecuencia de Palabras:

Este gráfico muestra cómo se distribuyen las frecuencias de las palabras clave en todo el conjunto de datos.

```
```R
hist(x = freq, main = "Histograma De Palabras Más Repetidas", xlab = "Frecuencia", ylab =
"",
col = "ivory")
```

```

## 3. Gráficos de las palabras más repetidas:

Estos gráficos muestran las palabras clave más frecuentes para las categorías "no desastre" y "desastre", respectivamente.

```
```R
plot(x = h1, main = "No Disaster Tweets", xlab = "Palabra", ylab = "Frecuencia", col =
"seagreen")
plot(x = h2, main = "Disaster Tweets", xlab = "Palabra", ylab = "Frecuencia", col =
"seagreen")
```

```

## ### 5. Análisis

- Categorización de Tweets: A partir de los cruces de datos, se identifica que hay palabras clave específicas que están predominantemente asociadas con tweets de desastre y otras con no desastre.
- Palabras clave: A través de las nubes de palabras y los gráficos de barras, se identifican las palabras clave más frecuentes. Por ejemplo, "body bags" es la palabra clave más común en la categoría de tweets que no tratan de desastres, mientras que "derailment" es predominante en tweets que sí tratan sobre desastres.

- Diferenciación: No parece haber solapamiento en las palabras clave más frecuentes entre las categorías de desastre y no desastre, lo que indica que esas palabras clave pueden ser muy descriptivas y útiles para categorizar y comprender el conjunto de datos.

Este análisis exploratorio utiliza gráficos visuales y frecuencias para comprender y diferenciar los tweets en función de si se refieren a un desastre o no. La utilización de palabras clave como indicadores puede ser valiosa para futuras aplicaciones, como la construcción de un modelo de clasificación.

## n-gramas

2. Creación de un corpus:

```
```r  
corpus <- Corpus(VectorSource(data$text))  
```
```

Un corpus es una colección de textos. En este caso, estamos tomando la columna 'text' de los datos y creando un corpus con ella. Esto nos permite trabajar con funciones específicas del paquete `tm` para analizar texto.

3. Definición del Tokenizer para bigramas:

```
```r  
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))  
```
```

Un tokenizer divide el texto en partes más pequeñas, como palabras o n-gramas. En este caso, estamos definiendo un tokenizer que se centra en bigramas (2-gramas).

4. Creación de la matriz de términos de documento para los bigramas:

```
```r  
tdm <- TermDocumentMatrix(corpus, control = list(tokenize = BigramTokenizer))  
```
```

Una matriz de términos de documento (TDM) muestra cuántas veces aparece cada bigrama en cada documento (o tweet, en este caso). El parámetro `tokenize = BigramTokenizer` indica que queremos dividir el texto en bigramas.

5. Convertir a matriz y obtener las frecuencias:

```
```r  
matrix <- as.matrix(tdm)  
bigram_freq <- sort(rowSums(matrix), decreasing = TRUE)  
```
```

Aquí, convertimos la TDM en una matriz regular y sumamos las apariciones de cada bigrama. Esto nos da la frecuencia total de cada bigrama en todos los tweets.

6. Convertir las frecuencias a un dataframe:

```
```r  
bigram_df <- data.frame(bigram = names(bigram_freq), frequency = bigram_freq)  
```
```

Creamos un dataframe para que sea más fácil trabajar con las frecuencias. Cada fila representa un bigrama y muestra su frecuencia.

7. Cálculo de las probabilidades:

```
```r  
total_bigrams <- sum(bigram_df$frequency)  
bigram_df$probability <- bigram_df$frequency / total_bigrams  
```
```

Aquí, estamos calculando la probabilidad de cada bigrama. La probabilidad es simplemente la frecuencia de ese bigrama dividida por el total de bigramas.

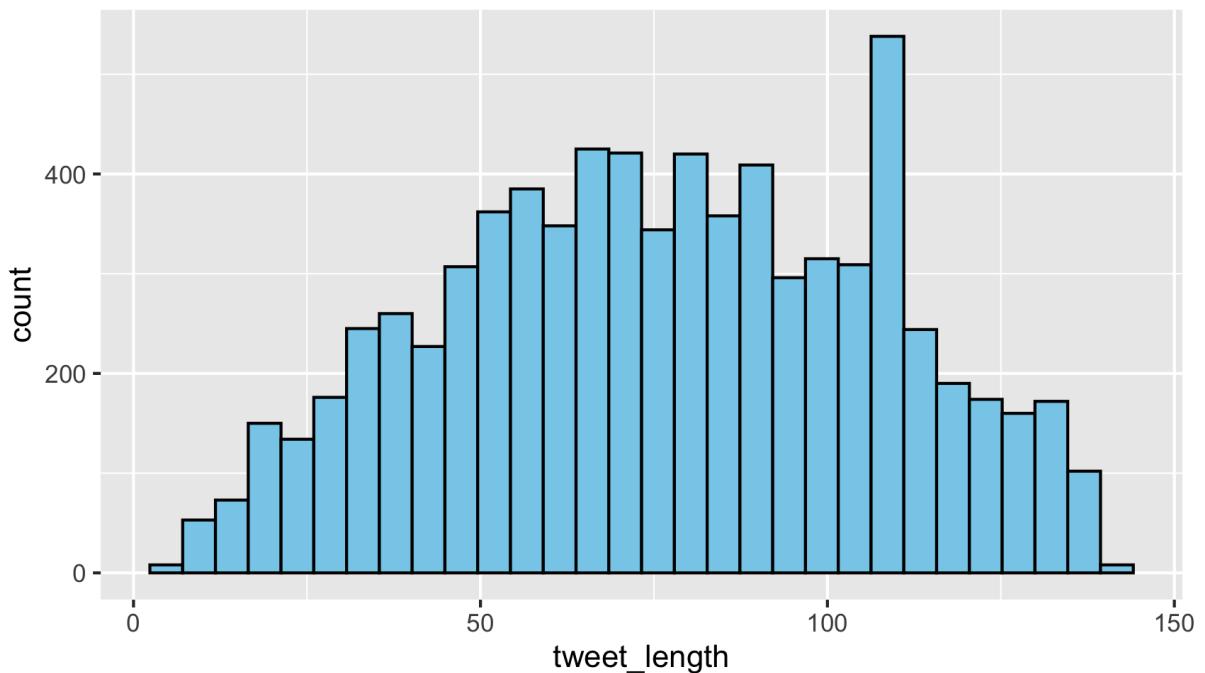
8. Mostrar los bigramas más comunes:

```
```r  
head(bigram_df)  
```
```

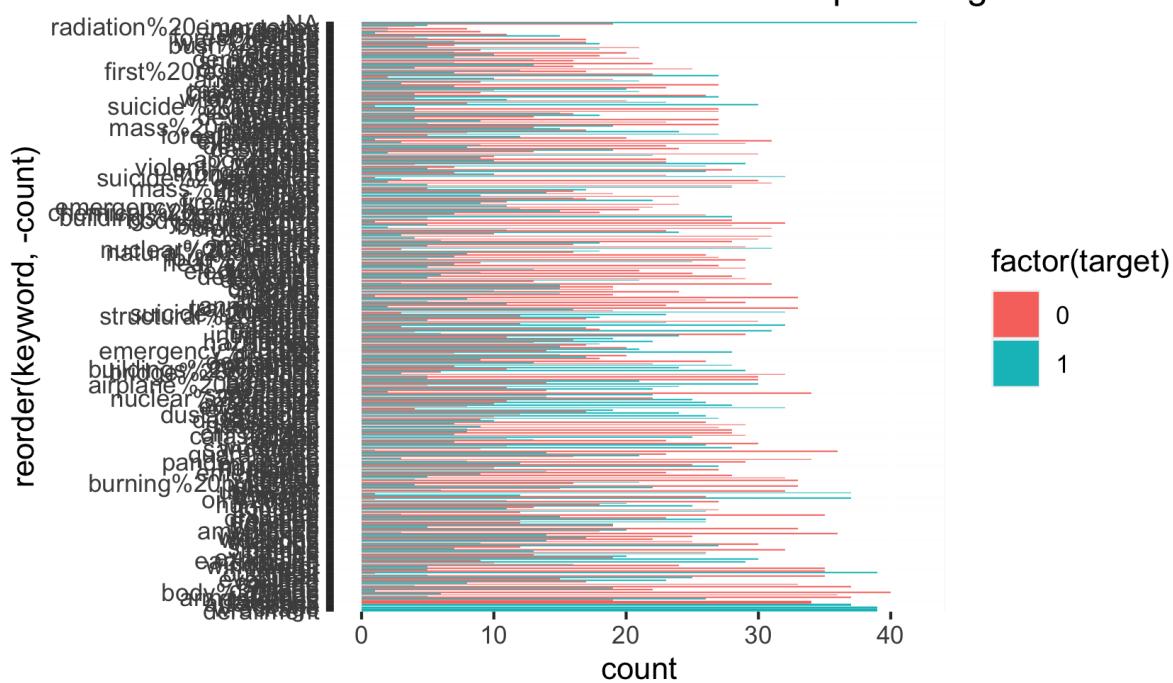
Esto es solo para visualizar los bigramas más frecuentes y sus probabilidades.

## Parte 5. Análisis exploratorio.

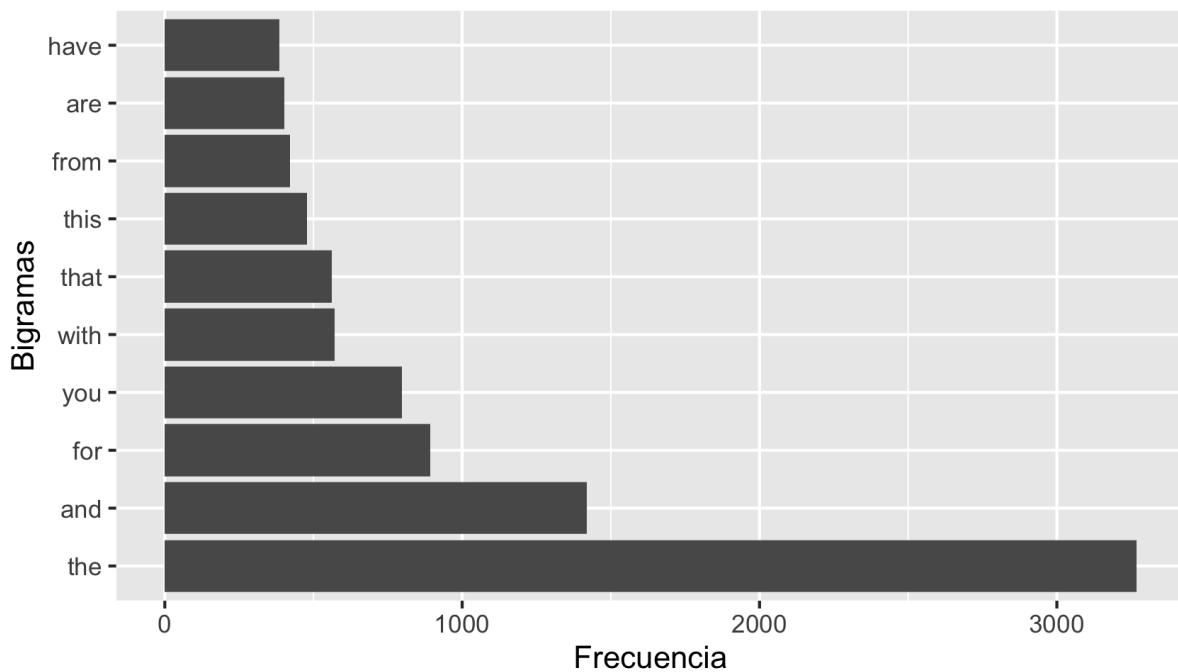
### Distribución de la longitud de tweets

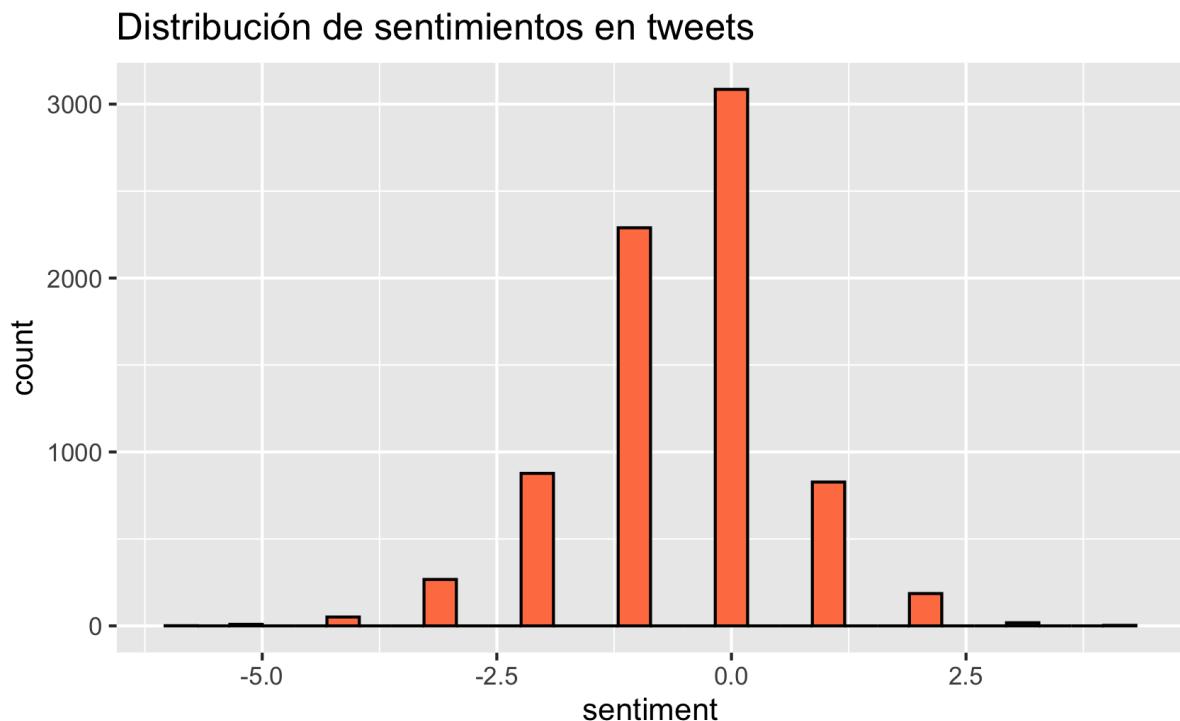


## Palabras clave más comunes por categoría



## Top 10 Bigramas





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

*detectarTweet.py*

#### 1. Importaciones:

- Se importan funciones y herramientas desde varias bibliotecas. Estas bibliotecas ayudan a trabajar con imágenes, matrices numéricas, gráficos, y más.
- Se importan específicamente herramientas para construir y entrenar modelos de redes neuronales usando TensorFlow y Keras.

#### 2. Cargando Datos:

- Se cargan dos archivos CSV: uno de entrenamiento ('train\_limpios.csv') y uno de prueba ('test\_limpios.csv').
- Se rellenan los valores faltantes en los datos con ceros.

#### 3. Preparación de Datos:

- Se separa la columna 'target' de los datos de entrenamiento para usarla como etiquetas (respuestas).
- Se convierten los datos en listas para su posterior uso.

#### 4. Construcción del Modelo:

- Se crea un modelo secuencial (un tipo de modelo neural que se define capa por capa en orden).
  - Se añaden tres capas al modelo:
    1. Capa densa con 12 neuronas y función de activación ReLU.

2. Capa densa con 8 neuronas y función de activación ReLU.
3. Capa densa con 1 neurona y función de activación Softmax (utilizada principalmente para problemas de clasificación).
  - Hay código comentado (líneas que comienzan con '#') que muestra otras formas de construir el modelo, pero no se ejecutan.

## 5. Compilación y Entrenamiento:

- Se compila el modelo. Aquí se define la función de pérdida (qué tan mal lo está haciendo el modelo) y el optimizador (cómo ajusta el modelo para mejorar).
- Se entrena el modelo usando los datos de entrenamiento (xTrain y yTrain) durante 150 épocas con un tamaño de lote de 10. Esto significa que el modelo verá y aprenderá de los datos 150 veces, ajustando sus pesos para mejorar su precisión.