

Universidad Tecnológica de la Mixteca

Ingeniería en Computación

Cómputo Flexible

Clasificador de Tweets sobre desastres naturales con red neuronal profunda

Armando Fabián González Valentín

Sumario

Introducción.....	4
Resumen.....	5
Problemática.....	6
Implementación.....	7
Conjunto de entrenamiento.....	7
Imports.....	7
Lectura del conjunto de datos.....	7
Visualización del conjunto de datos.....	9
Stopwords.....	12
Ngramas.....	15
Limpieza de los datos.....	15
Limpieza de URL's.....	15
Limpieza de etiquetas HTML.....	16
Limpieza de emojis.....	16
Datos limpios.....	17
Vectorización de los datos.....	17
División del Conjunto de datos.....	18
Construcción del modelo.....	19
Dropout.....	19
Función de activación ReLu.....	19
Función de activación Sigmoide.....	19
Aprendizaje.....	20
Compilación del modelo.....	20
Adam.....	20
Binary_crossentropy.....	20

Entropía cruzada.....	20
Entrenamiento.....	21
Predicción.....	22
.....	22
K-Folds.....	23
Análisis.....	23
Conclusión.....	27
Referencias.....	27

Introducción

El objetivo de este reporte es presentar la aplicación de una red neuronal a un problema real, es decir, una problemática que infiera en nuestro día a día como sociedad causando un perjuicio o entorpecimiento de procesos en las diferentes ramas de la industria y la ciencia.

Las redes neuronales tienen aplicaciones en ramas como la medicina con el análisis de imágenes médicas, la industria del software como lo son los clasificadores de spam o sistemas de recomendación como los utilizados en redes sociales como Facebook y Twitter por mencionar algunos.

Resumen

En las páginas siguientes se explicará el desarrollo e implementación de una red neuronal profunda para la clasificación de Tweets de noticias de desastres naturales, el objetivo de la red es clasificar como verdadera o falsa dicha noticia. Se hace una descripción detallada del procedimiento y la problemática que se intenta resolver, también se muestra el análisis de los resultados.

Problemática

Gran porcentaje de noticias que se reciben hoy en día se realizan a través de internet ya que es muy común recibir en nuestros teléfonos inteligentes una notificación sobre una noticia o algún tema en tendencia, pero tenemos un inconveniente con este sistema, el 70% de las noticias en internet son falsas debido a la facilidad con la que se propagan con tan solo dar un click.

Twitter es una red social de microblogueo o en pocas palabras es una red social donde podemos difundir noticias, ideas, apoyar algún movimiento social; con la característica que se cuenta con un número limitado de caracteres, y además no cuenta con censura de ningún tipo. Al no contar con un sistema de censura o de análisis de la información es posible difundir información falsa mediante Tweets.

Los rumores en Twitter se caracterizan por difundirse en forma de cascada, eso quiere decir que proceden de un origen común y se van haciendo cada vez más grandes al ser retuiteados(compartidos).

A continuación se muestra una grafica de la propagación de rumores según el tema o categoría.

Como se observa en la figura1 en primer lugar tenemos a las noticias falsas relacionadas a temas de política y en último lugar a las noticias relacionadas con desastres naturales.

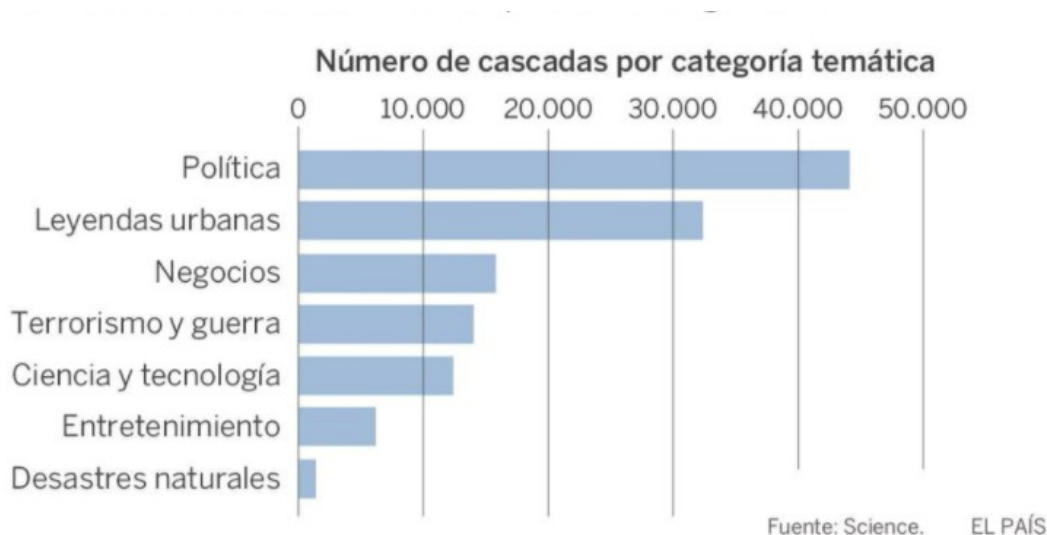


Figura 1: Tabla de propagación por tema

Para mostrar la aplicación de las redes neuronales hemos tomado el conjunto de datos de Tweets relacionados con desastres naturales, la razón de esto es que nuestro objetivo es mostrar la aplicación de una red neuronal a un problema real, pero no profundizar la solución óptima del problema, sin mencionar que solo se contó con el conjunto de datos relacionados a esta categoría.

Implementación

Lo primero que se realizó es el análisis de datos puesto que es una de las partes más importantes a la hora de plantear un modelo, esto nos permitirá el correcto planteamiento de la solución que se quiere dar y nos evitará ir a la deriva. El problema a tratar tiene que ver con el análisis de texto e identificar si lo que se escribe es real o falso, por lo tanto, debemos analizar el texto de la siguiente forma; extraer el número de palabras por Tweet, número de caracteres, palabras únicas por Tweet, Ngramas comunes y stopwords. Con estos datos se puede determinar si influyen en la veracidad de las publicaciones o que características se localizan en noticias falsas y de este modo clasificarlas.

Conjunto de entrenamiento

[Conjunto de datos](#)

El conjunto de datos de entrenamiento está dividido en Train y Test, el primer conjunto cuenta con 7613 elementos mientras que el segundo con 3263 elementos de los cuales ninguno cuenta con etiquetas a diferencia del primer conjunto.

Imports

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import numpy as np
import os
```

Figura 2 imports

Lectura del conjunto de datos

```
import pandas as pd

X_train = pd.read_csv('train.csv')
X_test = pd.read_csv('test.csv')

X_train
```

Figura 3: lectura del conjunto de datos

El conjunto del Train se organiza por 5 columnas: el id, palabras clave, locación, texto del Tweet, y la etiqueta de salida la cual marca con 1 las noticias verdaderas y con 0 las falsas.

index	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	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 being notified by officers. No other evacuation or shelter in place orders are expected	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation orders in California	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school	1
5	8	NaN	NaN	#RockyFire Update => California Hwy. 20 closed in both directions due to Lake County fire - #CAfire #wildfires	1
6	10	NaN	NaN	#flood #disaster Heavy rain causes flash flooding of streets in Manitou, Colorado Springs areas	1
7	13	NaN	NaN	I'm on top of the hill and I can see a fire in the woods...	1
8	14	NaN	NaN	There's an emergency evacuation happening now in the building across the street	1
9	15	NaN	NaN	I'm afraid that the tornado is coming to our area...	1
10	16	NaN	NaN	Three people died from the heat wave so far	1
11	17	NaN	NaN	Haha South Tampa is getting flooded hah- WAIT A SECOND I LIVE IN SOUTH TAMPA WHAT AM I GONNA DO WHAT AM I GONNA DO FVCK #flooding	1
12	18	NaN	NaN	#raining #flooding #Florida #TampaBay #Tampa 18 or 19 days. I've lost count	1
13	19	NaN	NaN	#Flood in Bago Myanmar #We arrived Bago	1
14	20	NaN	NaN	Damage to school bus on 80 in multi car crash #BREAKING	1
15	23	NaN	NaN	What's up man?	0
16	24	NaN	NaN	I love fruits	0
17	25	NaN	NaN	Summer is lovely	0
18	26	NaN	NaN	My car is so fast	0
19	28	NaN	NaN	What a goooooooooaaaaa!!!!!!	0

Tabla 4: Train

El conjunto del Test se organiza de la misma manera con la diferencia de que no cuenta con etiquetas de salida

index	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, stay safe everyone.
2	3	NaN	NaN	there is a forest fire at spot pond, geese are fleeing across the street, I cannot save them all
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan
5	12	NaN	NaN	We're shaking...It's an earthquake
6	21	NaN	NaN	They'd probably still show more life than Arsenal did yesterday, eh? EH?
7	22	NaN	NaN	Hey! How are you?
8	27	NaN	NaN	What a nice hat?
9	29	NaN	NaN	Fuck off!
10	30	NaN	NaN	No I don't like cold!
11	35	NaN	NaN	NOOOOOOOOO! Don't do that!
12	42	NaN	NaN	No don't tell me that!
13	43	NaN	NaN	What if?!

Tabla 5: Test

Visualización del conjunto de datos

```
print("Tamaño del Train: ", len(X_train))  
print("Tamaño del Test: ", len(X_test))
```

```
Tamaño del Train: 7613  
Tamaño del Test: 3263
```

Figura 6: Tamaño del conjunto

En el conjunto de entrenamiento contamos con 4342 noticias falsas y 3271 noticias verdaderas

```
# Tweets falsos y verdaderos en Train  
X_train['target'].value_counts()
```

```
0    4342  
1    3271
```

Figura 7: casos verdaderos y falsos

En la figura 7 observamos la distribución de las noticias falsas y verdaderas en el conjunto de entrenamiento.

```
X_train['target'].hist()  
plt.ylabel("Número tweets")  
plt.show()
```

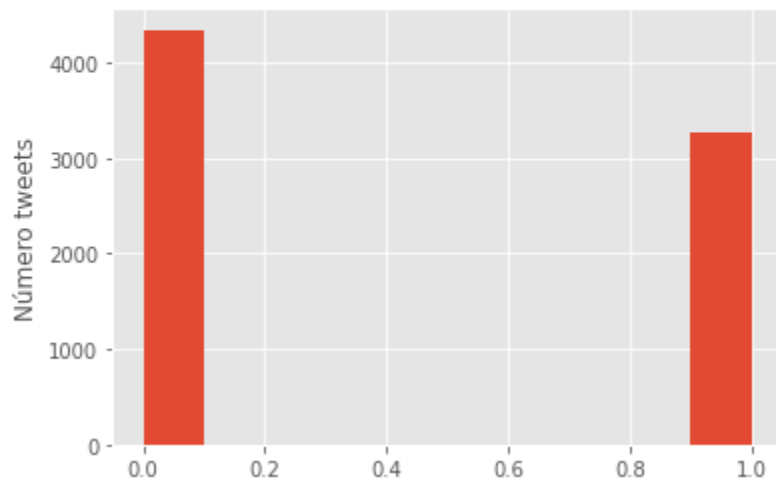


Figura 7: Distribución de los datos en el Train

Vamos a explorar los datos para determinar el modelo a seguir con el promedio de palabras por Tweet, esto nos puede dar una idea si la longitud de la noticia está relacionada con la veracidad de la misma.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
tweet_falso = X_train[X_train['target'] == 0]['text'].str.split().map(lambda x: len(x))
tweet_real = X_train[X_train['target'] == 1]['text'].str.split().map(lambda x: len(x))

ax1.hist(tweet_falso, color='red')
ax1.set_title('Noticias Reales')
ax2.hist(tweet_real, color='blue')
ax2.set_title('Noticias falsas')
fig.suptitle('Número de palabras por tweet')
plt.show()
```

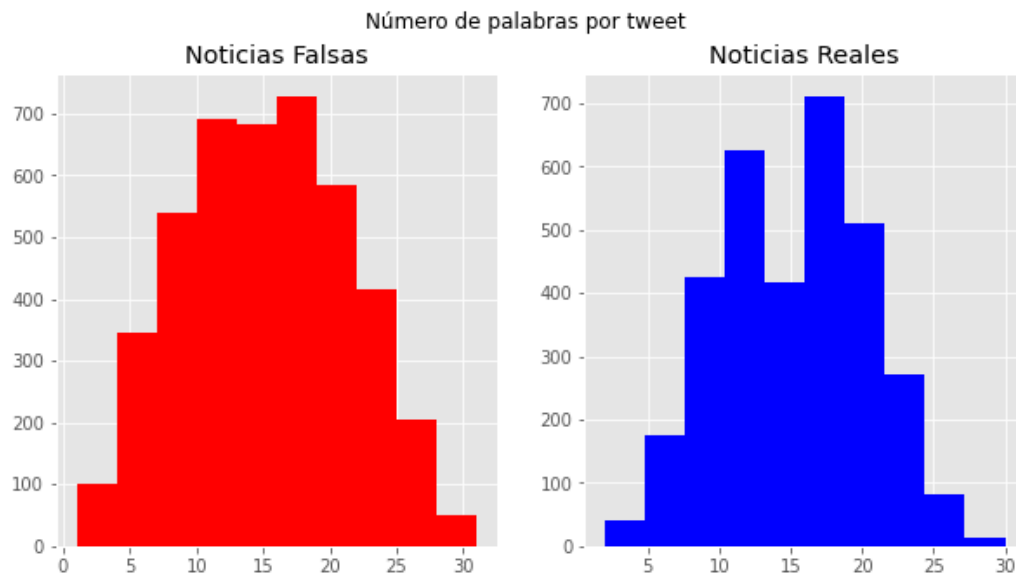


Figura 8: Distribución media de palabras por Tweet

Se observa que para noticias reales el promedio se encuentra entre 15 y 20 palabras mientras que para las noticias falsas tenemos un rango de 10 a 20 palabras, el promedio en ambos casos es similar por lo que no se puede asegurar que influye en la veracidad.

Debemos tomar en cuenta otros parámetros como el promedio de palabras únicas por Tweet, la longitud media de palabras por Tweet, y el número de caracteres por Tweet.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))

tweet_falso = X_train[X_train['target'] == 0]['text'].str.split().map(lambda x: len(set(x)))
tweet_real = X_train[X_train['target'] == 1]['text'].str.split().map(lambda x: len(set(x)))

ax1.hist(tweet_falso, color='red')
ax1.set_title('Noticias Falsas')
ax2.hist(tweet_real, color='blue')
ax2.set_title('Noticias Reales')

fig.suptitle('Número de palabras únicas por tweet')
plt.show()
```

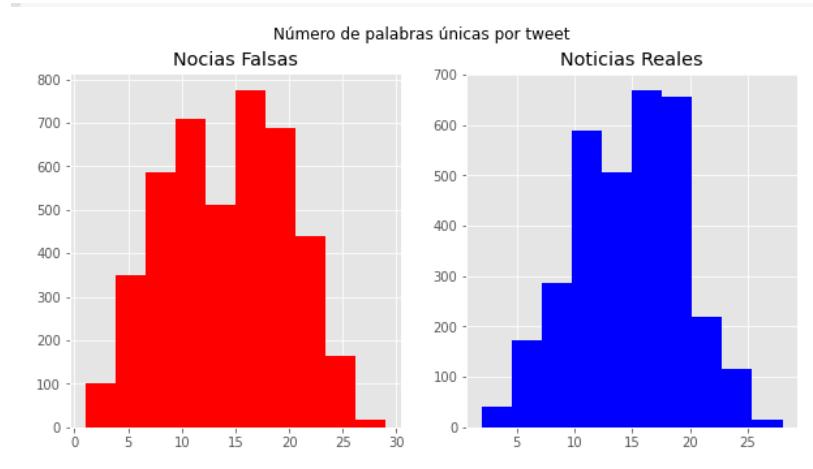


Figura 9: promedio de palabras únicas por Tweet

De igual manera los promedios son similares en ambas gráficas.

La longitud por palabra oscila entre 5 y 7.5 para noticias verdaderas a diferencia de las noticias falsas que tienen una longitud de 5.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))

tweet_falso = X_train[X_train['target'] == 0]['text'].str.split().map(lambda x: np.mean([len(i) for i in x]))
tweet_real = X_train[X_train['target'] == 1]['text'].str.split().map(lambda x: np.mean([len(i) for i in x]))

ax1.hist(tweet_falso, color='red')
ax1.set_title('Noticias Falsas')

ax2.hist(tweet_real, color='blue')
ax2.set_title('Noticias Reales')

fig.suptitle('Longitud media de las palabras por Tweet')

plt.show()
```

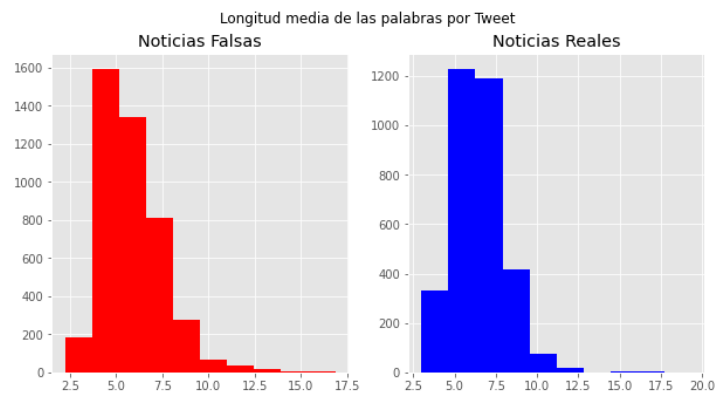


Figura 10: longitud media por palabra de Tweet

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))

# Calculamos el número de caracteres por tweet
tweet_falso = X_train[X_train['target'] == 0]['text'].str.len()
tweet_real = X_train[X_train['target'] == 1]['text'].str.len()

ax1.hist(tweet_falso, color='red')
ax1.set_title('Noticias Falsas')

ax2.hist(tweet_real, color='blue')
ax2.set_title('Noticias Reales')

fig.suptitle('Número de caracteres por tweet')

plt.show()
```

El número de caracteres para los dos casos está entre 125 y 140 caracteres.

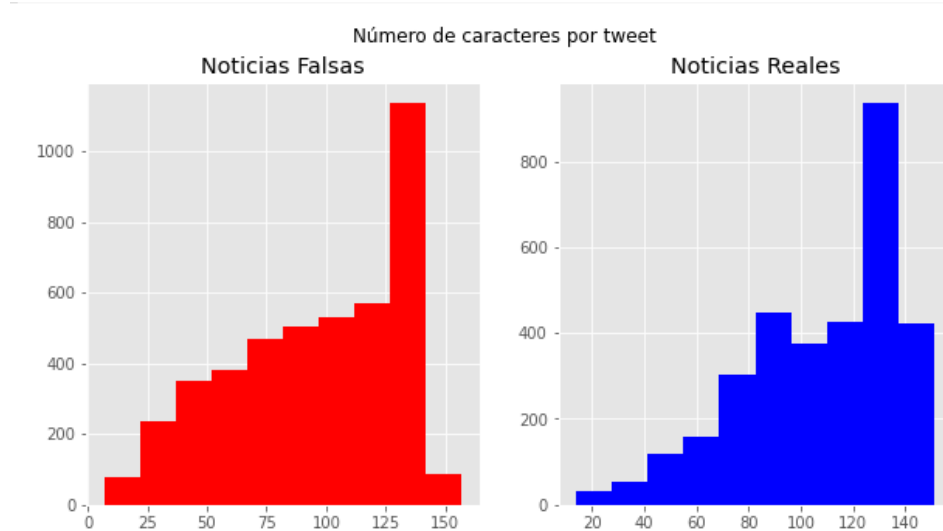


Figura 11: número de caracteres por Tweet

Stopwords

El siguiente paso luego de extraer características del texto es extraer las palabras más comunes en este tipo de publicaciones, a estas palabras se les conoce como stopwords.

Estas palabras no tienen un significado por si solas, sino que modifican o acompañan a otras, este grupo suele estar conformado por artículos, pronombres, preposiciones, adverbios e incluso algunos verbos.

En el procesamiento de datos en lenguaje natural son filtradas antes o después del proceso en si, no se consideran por su nulo significado, en el caso de los buscadores como Google no lo consideran al momento de posicionar, pero si al momento de mostrar los resultados de búsqueda.

Lo que se realizó a continuación es obtener un conjunto de palabras más usadas en el idioma Inglés para su comparación con los Tweets para así saber cuales son las palabras más usadas en las publicaciones.

La siguiente biblioteca nos ayuda a obtener las palabras más usadas en distintos idiomas.

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
stopwords.words('english')
```

Se define una función para comparar las palabras utilizadas en las publicaciones con las más usadas en el idioma, de este modo podemos identificar cuales son las que se suelen usar en noticias falsas.

```
def stopwords_promedio(clase):
    tweets_stopwords = {}
    for palabras in X_train[X_train['target'] == clase]['text'].str.split():
        sw = list(set(palabras).intersection(stopwords.words('english')))
        for i in sw:
            if i in tweets_stopwords.keys():
                tweets_stopwords[i] += 1
            else:
                tweets_stopwords[i] = 1

    top = sorted(tweets_stopwords.items(), key=lambda x:x[1],reverse=True)[:10]
    plt.bar(*zip(*top))
    plt.show()
```

```
stopwords_promedio(0)
```

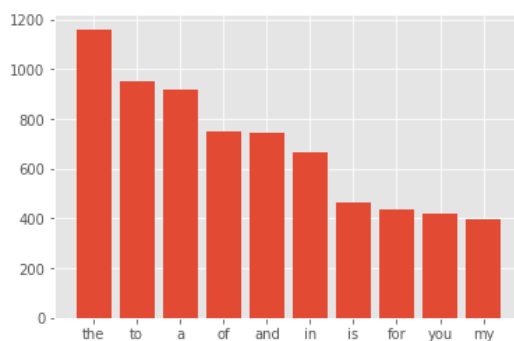


Figura 12: stopwords en noticias falsas

Palabras más usadas en noticias reales y noticias falsas suelen ser las mismas con la particularidad que cambian de frecuencia.

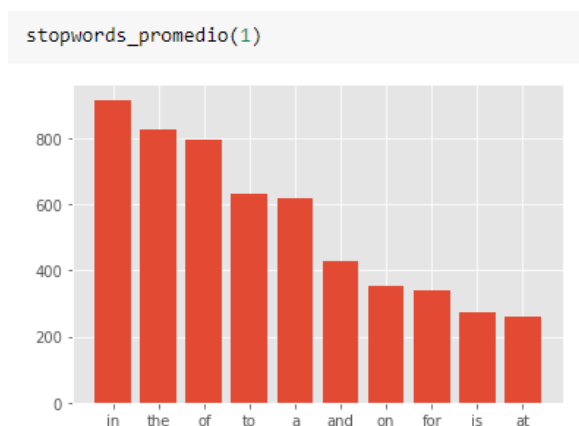


Figura 13: stopwords en noticias reales

Hacemos el mismo análisis para los signos de puntuación

Signos de puntuación en noticias falsas

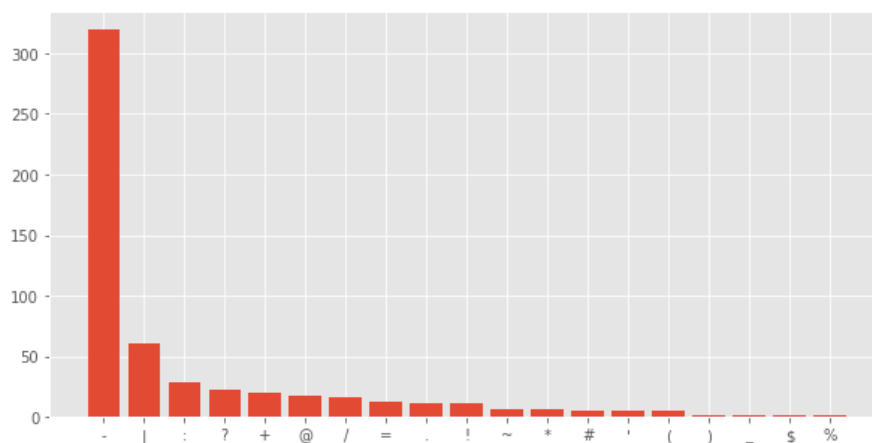


Figura 14: signos de puntuación en noticias falsas

Signos de puntuación en noticias verdaderas

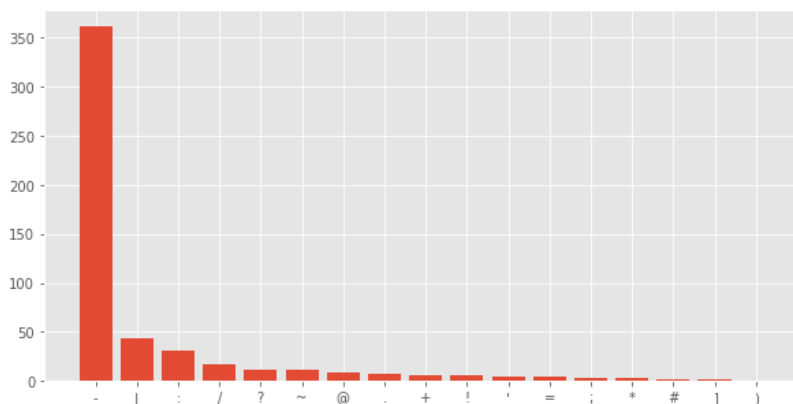


Figura 15: signos de puntuación en noticias verdaderas

Ngramas

Un n-grama es una subsecuencia de n elementos de una secuencia dada.

Ahora vamos a analizar los n-gramas que aparecen en las publicaciones, en este caso analizaremos bigramas con ayuda de la biblioteca de sklearn.

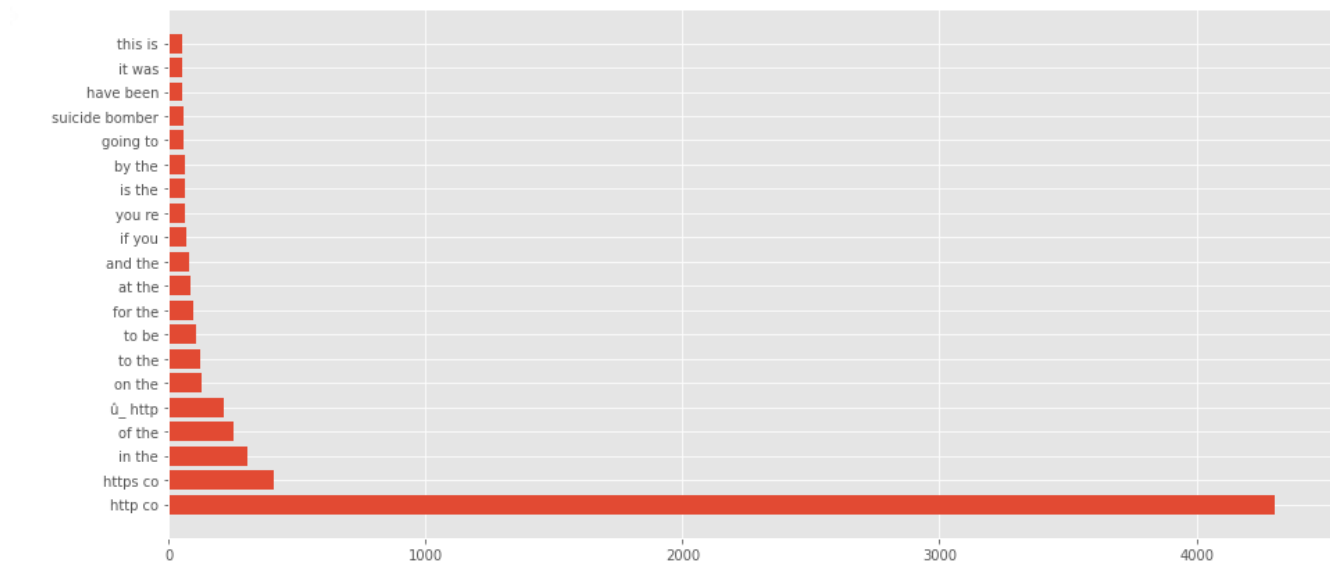


Figura 16: bigramas en Tweets

Podemos observar que en casi todos los Tweets aparecen links o URL's, estos elementos no significan nada en cuanto a la veracidad del texto por lo que podemos eliminar aquellos elementos irrelevantes.

Limpieza de los datos

Aquellos elementos que no toman importancia se pueden omitir y limpiar, una forma de hacerlo es procesar el texto con expresiones regulares.

Limpieza de URL's

Creamos una función para limpiar el texto de URL's [Expresiones regulares](#)

```
import re

def limpia_url(texto):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'', texto)

limpia_url("Computo flexible: https://www.utm.mx/")

'Computo flexible: '
```

Figura 17: limpieza de url's

Limpieza de etiquetas HTML

Creamos una función para limpiar las etiquetas HTML del texto

```
from html.parser import HTMLParser

class HTMLparser(HTMLParser):
    def __init__(self):
        self.reset()
        self.strict = False
        self.convert_charrefs = True
        self.fed = []

    def handle_data(self, d):
        self.fed.append(d)

    def get_data(self):
        return ''.join(self.fed)

def limpia_html(texto):
    s = HTMLparser()
    s.feed(texto)
    return s.get_data()

limpia_html('<html><head><title>Test</title></head>'
            '<body><h1>Parse me!</h1></body></html>')

'TestParse me!'
```

Figura 18: limpieza de etiquetas HTML

Limpieza de emojis

```
def limpia_emoji(texto):
    emoji_patron = re.compile("[\n
        u\"\\U0001F600-\\U0001F64F\" # emoticones
        u\"\\U0001F300-\\U0001F5FF\" # symbols & pictographs
        u\"\\U0001F680-\\U0001F6FF\" # transport & map symbols
        u\"\\U0001F1E0-\\U0001F1FF\" # flags (iOS)
        u\"\\U00002702-\\U000027B0\"
        u\"\\U000024C2-\\U0001F251\"
        \"]+", flags=re.UNICODE)

    return emoji_patron.sub(r'', texto)

limpia_emoji("No se gresa a clases por covid | 🤔🙄")

'No se gresa a clases por covid '
```

Figura 19: limpieza de emojis

Por último limpiamos los signos de puntuación ya que tampoco aportan información importante al texto.


```
def limpieza_puntuacion(texto):
    return texto.translate(str.maketrans('', '', string.punctuation))

limpieza_puntuacion("#odio, la. uni")

'odio la uni'
```

Figura 20: limpieza de signos de puntuación

Datos limpios

Se aplica la limpieza al conjunto de datos Train y Test, como se observa en la figura 21 ya no hay datos basura como URL's, signos, emojis etc.

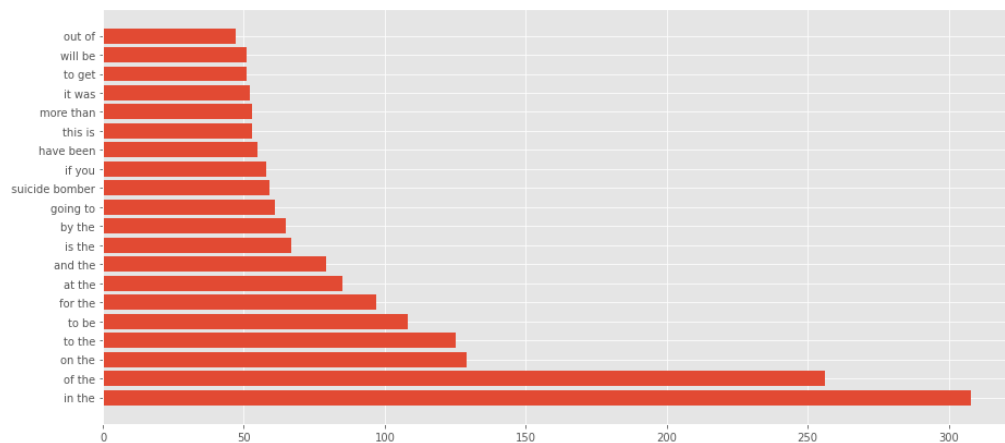


Figura 21: Datos limpios

Vectorización de los datos

Un paso fundamental que se debe realizar cuando se está analizando texto es la vectorización, la cual consiste en convertir las palabras en arreglos de números.

```
Y_train = X_train_prep['target']

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train_prep['text'])

X_train = X_train.toarray()
```

Figura 22: vectorización train

```
X_test = vectorizer.transform(X_test_prep['text'])
X_test = X_test.toarray()
```

Figura 23: vectorización test

Podemos notar una diferencia en el conjunto Train y Test a la hora de vectorizar, a uno se le aplica el metodo **fit_transform** y al otro solo **transform**. Esto se debe a que **fit** calcula las medias de las columnas a partir de algunos datos, y **transform** aplica esas medias a algunos datos (que simplemente reemplaza los valores faltantes con los medias). Si ambos datos son iguales (es decir, los datos para calcular las medias y los datos a los que se aplican las medias), puede utilizarlos, **fit_transform** que es básicamente un **fit** seguido de **transform**.

En términos simples, **fit_transform** significa hacer algunos cálculos y luego hacer la transformación (por ejemplo, calcular el promedio de las columnas a partir de algunos datos y luego reemplazar los valores faltantes). Entonces, para el conjunto de entrenamiento, debe calcular y hacer la transformación. Pero para el conjunto de prueba, el aprendizaje automático aplica la predicción en función de lo aprendido durante el conjunto de entrenamiento y, por lo tanto, no necesita calcular, solo realiza la transformación.

División del Conjunto de datos

Como hemos estado haciendo cambios al conjunto de datos al momento de limpiar el texto que no es relevante debemos hacer una nueva división del conjunto.

```
from sklearn.model_selection import train_test_split
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.15)
```

Figura 24: división del conjunto de datos

Usamos el 15% para el test

El nuevo tamaño de cada conjunto es el siguiente

```
print("tamaño entrenamiento: ", len(X_train))
print("tamaño de validación: ", len(X_val))
print("tamaño pruebas: ", len(X_test))

tamaño entrenamiento: 6471
tamaño de validación: 1142
tamaño pruebas: 3263
```

Figura 25: Conjuntos finales

Construcción del modelo

Para construcción del modelo hemos implementado una red neuronal de tres capas, una capa de entrada con 16 neuronas, una capa oculta con 16 neuronas, y una capa de salida con 1 neurona. También dos capas intermedias de dropout para reducir el sobre entrenamiento u overfitting.

Las dos primeras capas tienen una función de activación **relu** y la última capa una función sigmoide, de acuerdo a la literatura consultada por nomatividad en problemas de clasificación binaria se emplea la función **relu** para las capas de entrada y las hidden layer, sigmoide para la capa de salida.

Dropout

Dropout es un metodo que desactiva un numero de neuronas de una red neuronal de forma aleatoria. En cada iteración de la red neuronal dropout desactivara diferentes neuronas, las neuronas desactivadas no se toman en cuenta para el **forwardpropagation** ni para el **backwardpropagation** lo que obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas. Este método ayuda a reducir el **overfitting** ya que las neuronas cercanas suelen aprender patrones que se relacionan y estas relaciones pueden llegar a formar un patrón muy especifico con los datos de entrenamiento, con dropout esta dependencia entre neuronas es menor en toda la red neuronal, de esta manera la neuronas necesitan trabajar mejor de forma solitaria y no depender tanto de las relaciones con las neuronas vecinas.

Dropout tiene un parámetro que indica la probabilidad de que las neuronas se queden activadas, este parámetro toma valores de 0 a 1, 0.5 suele usarse por defecto indicando que la mitad de las neuronas se quedaran activadas, si los valores son cercanos a 0 dropout desactivara menos neuronas, si es cercano a 1 desactivara muchas más neuronas. Dropout solo se usa durante la fase de entrenamiento, en la fase de pruebas ninguna neurona se desactiva pero las escalamos con la probabilidad del dropout para compensar a las neuronas desactivadas durante la fase de entrenamiento.

Función de activación ReLu

La Unidad lineal rectificada es la función de activación más utilizada en los modelos de aprendizaje profundo. La función devuelve 0 si recibe una entrada negativa, pero para cualquier valor positivo x devuelve ese valor. Entonces se puede escribir como **$f(x)=\max(0,x)$** .

Función de activación Sigmoide

La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0.

$$f(x) = \frac{1}{1 + e^{-x}}$$

En la imagen de abajo se aprecia el modelo, a las capas de dropout se les asignó una probabilidad de 40% para mantener la neuronas en cada capa.

```
from tensorflow.keras import models
from tensorflow.keras import layers

model = models.Sequential()

model.add(layers.Dense(16, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(1, activation='sigmoid'))
```

Figura 26: modelo de la red

Aprendizaje

La red implementada hace uso del modelo secuencial de Tensorflow, es apropiado para una pila de capas, donde cada capa tiene exactamente un tensor de entrada y un tensor de salida.

Un tensor es un **vector** o **matriz** de n-dimensiones que representa todos los tipos de datos. Todos los valores de un tensor contienen un tipo de datos idéntico con una forma **conocida** (o parcialmente conocida). La forma de los datos es la dimensionalidad de la matriz.

Compilación del modelo

El modelo utiliza el optimizador **adam**, la función de pérdida o de error **binary_crossentropy**, y las métricas a medir son la **accuracy** y **precision**.

Adam

Adaptative Moment Optimization combina la metodología de Momentum y RMSProp, calculando una combinación lineal entre el gradiente y el incremento anterior, y considera los gradientes recientemente aparecidos en las actualizaciones para mantener diferentes tasas de aprendizaje por variable.

Binary_crossentropy

calcula la pérdida de entropía cruzada entre etiquetas verdaderas y etiquetas predichas.

Entropía cruzada

La entropía cruzada es una métrica que puede utilizarse para reflejar la precisión de los pronósticos probabilísticos y está estrechamente vinculada con la estimación por máxima verosimilitud.

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy', 'Precision'])
)
```

Figura 27: compilación del modelo

Entrenamiento

Para el entrenamiento se fijaron 20 épocas de iteración, **batch_size** de 1024, y **validation_data**; datos sobre los que evaluar la pérdida y cualquier métrica del modelo al final de cada época.

```
history = model.fit(
    X_train,
    Y_train,
    epochs=20,
    batch_size=1024,
    validation_data=(X_val, Y_val))
```

Figura 28: entrenamiento

Una vez entrenado el modelo obtenemos el siguiente resultados, una precisión de 92% y una exactitud de 92%, lo que nos indica que el modelo está clasificando de forma correcta pero un porcentaje muy alto también muestra sobre entrenamiento, una característica del modelo secuencial de keras es que toma el historial de la última época y la toma como referencia si se vuelve a entrenar el modelo.

1s 101ms/step - loss: 0.3013 - accuracy: 0.9295 - precision: 0.9246 - val_loss: 0.4639 - val_accuracy: 0.7907 - val_precision: 0.7689

Figura 29: métrica de modelo

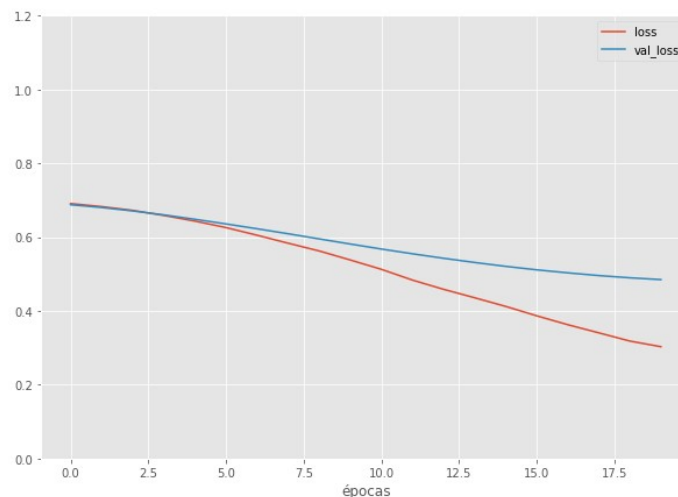
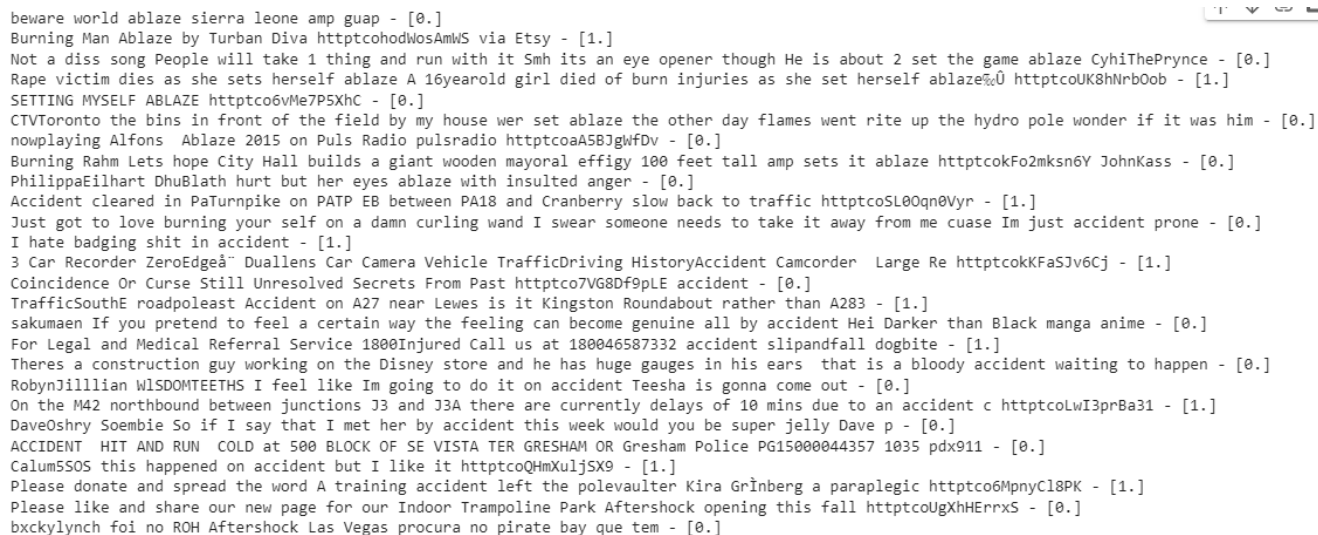


Figura 30: gráfica de error

En la gráfica anterior nos muestra el error que se está produciendo en el conjunto de validación y esto nos indica que existe sobre-entrenamiento.

Predicción

Probando el modelo obtenemos la siguiente clasificación



```
beware world ablaze sierra leone amp guap - [0.]
Burning Man Ablaze by Turban Diva httpcohodWosAmwS via Etsy - [1.]
Not a diss song People will take 1 thing and run with it Smh its an eye opener though He is about 2 set the game ablaze CyhiThePrynce - [0.]
Rape victim dies as she sets herself ablaze A 16yearold girl died of burn injuries as she set herself ablaze%Ü httpcoUK8hNrbOob - [1.]
SETTING MYSELF ABLAZE httpco6vMe7P5XhC - [0.]
CTVToronto the bins in front of the field by my house wer set ablaze the other day flames went rite up the hydro pole wonder if it was him - [0.]
nowplaying Alfons Ablaze 2015 on Puls Radio pulsradio httpcoaA5BJgWfDv - [0.]
Burning Rahm Lets hope City Hall builds a giant wooden mayoral effigy 100 feet tall amp sets it ablaze httptcokFo2mksn6Y JohnKass - [0.]
PhilippaEilhart DhuBlath hurt but her eyes ablaze with insulted anger - [0.]
Accident cleared in PaTurnpike on PATP EB between PA18 and Cranberry slow back to traffic httpcoSL00qn0Vyr - [1.]
Just got to love burning your self on a damn curling wand I swear someone needs to take it away from me cuase Im just accident prone - [0.]
I hate badging shit in accident - [1.]
3 Car Recorder ZeroEdgeâ DualLens Car Camera Vehicle TrafficDriving HistoryAccident Camcorder Large Re httptcokKFasJv6Cj - [1.]
Coincidence Or Curse Still Unresolved Secrets From Past httpco7VG8Df9pLE accident - [0.]
TrafficSouthE roadpoleast Accident on A27 near Lewes is it Kingston Roundabout rather than A283 - [1.]
sakumaen If you pretend to feel a certain way the feeling can become genuine all by accident Hei Darker than Black manga anime - [0.]
For Legal and Medical Referral Service 1800Injured Call us at 180046587332 accident slipandfall dogbite - [1.]
Theres a construction guy working on the Disney store and he has huge gauges in his ears that is a bloody accident waiting to happen - [0.]
RobynJillillian WLSDOMTEETHS I feel like Im going to do it on accident Teesha is gonna come out - [0.]
On the M42 northbound between junctions J3 and J3A there are currently delays of 10 mins due to an accident c httpcoLwI3prBa31 - [1.]
DaveOshry Soembie So if I say that I met her by accident this week would you be super jelly Dave p - [0.]
ACCIDENT HIT AND RUN COLD at 500 BLOCK OF SE VISTA TER GRESHAM OR Gresham Police PG15000044357 1035 pdx911 - [0.]
Calum550S this happened on accident but I like it httpcoQHmXulj5X9 - [1.]
Please donate and spread the word A training accident left the polevaulter Kira GrInberg a paraplegic httpco6MpnYCl8PK - [1.]
Please like and share our new page for our Indoor Trampoline Park Aftershock opening this fall httpcoUgXhHErrxS - [0.]
bxckylynch foi no ROH Aftershock Las Vegas procura no pirate bay que tem - [0.]
```

Figura 31: clasificación

El modelo está clasificando correctamente aunque tiene un margen de error, notamos en el resultado que aún sigue filtrando algunos URL's, de cualquier forma se logró el objetivo de crear un clasificador de Tweets.

K-Folds

Ahora haciendo uso del método K-Folds haremos el entrenamiento de la red, configurando los folds de la siguiente manera. El conjunto de entrenamiento se divide en 2 folds, el estado random no es inicializado, y shuffle se asigna a False, a la red se le inicializó con 10 iteraciones.

```
from sklearn.model_selection import KFold
|
X= X_train
y = Y_train
hf=KFold(n_splits= 2, random_state=None, shuffle=False)
for train_index, test_index in kf.split(X ):
    X_entrenamiento, X_prueba = X[train_index], X[test_index]
    Y_entrenamiento, Y_prueba = y.iloc[train_index], y.iloc[test_index]
    history = model.fit(
        X_entrenamiento,
        Y_entrenamiento,
        epochs=10,
        batch_size=1024,
        validation_data=(X_prueba, Y_prueba))
    print(f'\nTrain{train_index}      Test{test_index}\n')
```

Análisis

Al entrenar la red obtenemos los siguientes resultados en la última iteración de cada fold:

iteración

Primer

```
Epoch 10/10
5/5 [=====] - 1s 112ms/step - loss: 0.6130 - accuracy: 0.7250 - precision: 0.9889 - val_loss: 0.6396 - val_accuracy: 0.6754
```

Segunda

```
Epoch 10/10
5/5 [=====] - 1s 118ms/step - loss: 0.4424 - accuracy: 0.8747 - precision: 0.9747 - val_loss: 0.4662 - val_accuracy: 0.8555
```

Tercera

```
Epoch 10/10
5/5 [=====] - 1s 118ms/step - loss: 0.2823 - accuracy: 0.9385 - precision: 0.9726 - val_loss: 0.2837 - val_accuracy: 0.9212
```

Cuarta

Epoch 10/10

5/5 [=====] - 1s 114ms/step - loss: 0.1711 - accuracy: 0.9652 - precision: 0.9770 - val_loss: 0.1457 - val_accuracy: 0.9719

Quinta

Epoch 10/10

5/5 [=====] - 1s 117ms/step - loss: 0.1106 - accuracy: 0.9810 - precision: 0.9850 - val_loss: 0.0717 - val_accuracy: 0.9869

Sexta

Epoch 10/10

5/5 [=====] - 1s 113ms/step - loss: 0.0771 - accuracy: 0.9862 - precision: 0.9885 - val_loss: 0.0420 - val_accuracy: 0.9925

Séptimo

Epoch 10/10

5/5 [=====] - 1s 116ms/step - loss: 0.0550 - accuracy: 0.9921 - precision: 0.9911 - val_loss: 0.0440 - val_accuracy: 0.9906

Octava

Epoch 10/10

5/5 [=====] - 1s 116ms/step - loss: 0.0437 - accuracy: 0.9915 - precision: 0.9911 - val_loss: 0.0214 - val_accuracy: 0.9944

Noveno

Epoch 10/10

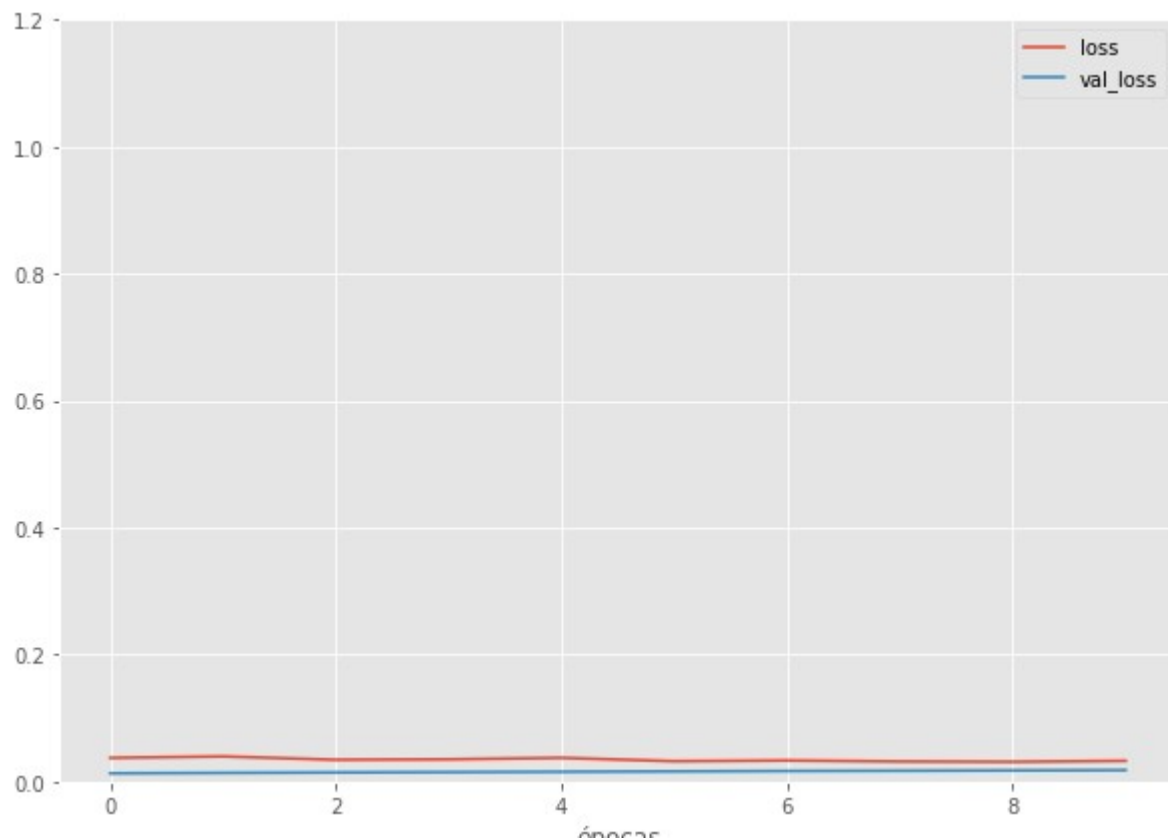
5/5 [=====] - 1s 119ms/step - loss: 0.0372 - accuracy: 0.9935 - precision: 0.9917 - val_loss: 0.0284 - val_accuracy: 0.9887

Décima

Epoch 10/10

5/5 [=====] - 1s 117ms/step - loss: 0.0326 - accuracy: 0.9937 - precision: 0.9951 - val_loss: 0.0179 - val_accuracy: 0.9944

En cada iteración podemos observar como la precisión y la exactitud aumentan, el error con respecto al conjunto de entrenamiento y el conjunto de validación decrece en cada iteración. Como consecuencia de esto tenemos la grafica del error que ha disminuido de manera considerable llegando casi al 0 y se ha mantenido lineal.



Con esta gráfica podemos afirmar que nuestra red está realizando una clasificación aun mejor, es hora de realizar una predicción con el conjunto de datos no etiquetado para verificar lo antes mencionado.

```
Y_pred = model.predict(X_test).round(0)

for i in range(60):
    print("{} - {}".format(X_test_prep['text'][i], Y_pred[i]))
```

En la imagen de abajo se muestran las clasificaciones que ha hecho la red una vez que ha entrenado con K-Folds, se ha mejorado casi al 100% la precisión y la exactitud. Las frases cortas que claramente no

son noticias sino comentarios comunes son clasificados en 0 como falsos, pero aquellos que en su contenido hacen referencia a desastres naturales son puestos a 1 como verdaderos.

Just happened a terrible car crash - [0.]
Heard about earthquake is different cities stay safe everyone - [1.]
there is a forest fire at spot pond geese are fleeing across the street I cannot save them all - [1.]
Apocalypse lighting Spokane wildfires - [1.]
Typhoon Soudelor kills 28 in China and Taiwan - [1.]
Were shakingIts an earthquake - [1.]
Theyd probably still show more life than Arsenal did yesterday eh EH - [0.]
Hey How are you - [0.]
What a nice hat - [0.]
Fuck off - [0.]
No I dont like cold - [0.]
NOOOOOOOOO Dont do that - [0.]
No dont tell me that - [0.]
What if - [0.]
Awesome - [0.]

Antes del método K-Folds se tenía el problema que la red no discriminaba de forma correcta las noticias con algún link, ahora la red diferencia entre noticias que tienen links con contenido sobre algún evento natural o aquellos que no lo tienen.

La primer noticia claramente es una noticia real, pero tiene un link, esto reitera que la red la está clasificando adecuadamente. Observemos la penúltima noticia, solo contiene una frase y un enlace pero claramente no es una noticia real, la red lo ha clasificado como falsa.

Birmingham Wholesale Market is ablaze BBC News Fire breaks out at Birminghams Wholesale Market httpcoirWqCEZWEU - [1.]
sunkxssedharry will you wear shorts for race ablaze - [0.]
PreviouslyOnDoyinTv Toke Makinwa's marriage crisis sets Nigerian Twitter ablaze httpcoCMghxBa2XI - [1.]
Check these out httpcorOI2NSmEJJ httpco3Tj8ZjiN21 httpcoYDUiXEfIpE httpcoLxTjc87KLS nsfw - [0.]
PSA I'm splitting my personalities

Conclusión

Con respecto al problema se concluye que el modelo obtenido es funcional , después de aplicar el método K-Folds se mejoró un 7% más, lo cual es suficiente para asegurar un 98.57%-99.68 % de efectividad.

Referencias

[Neuralnetworksanddeeplearning](#)

https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/32/pdf

[Procesamiento de lenguaje natural y sus aplicaciones-](#)

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

<https://unipython.com/como-preparar-datos-de-texto-con-scikit-learn/>

<https://naps.com.mx/blog/vectorizar-texto-para-machine-learning/>

<https://scikit-learn.org/stable/modules/generated/>

[sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer.fit_transform](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer.fit_transform)

https://keras.io/api/losses/probabilistic_losses/#binary_crossentropy-class

<https://www.lokad.com/es/definicion-de-entropia-cruzada#:~:text=La%20entrop%C3%ADa%20cruzada%20es%20una,la%20estimaci%C3%B3n%20por%20m%C3%A1xima%20verosimilitud.>

<https://keras.rstudio.com/reference/fit.html>

<https://qastack.mx/datascience/12321/difference-between-fit-and-fit-transform-in-scikit-learn-models#:~:text=En%20t%C3%A9rminos%20simples%2C%20fit%20transform%20significa,calcular%20y%20hacer%20la%20transformaci%C3%B3n.>

https://www.tensorflow.org/guide/keras/sequential_model

<https://guru99.es/tensor-tensorflow/>

<https://realpython.com/python-keras-text-classification/>

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

<https://vincentblog.xyz/posts/dropout-y-batch-normalization>

<https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning>

<https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/>

<https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/>

<https://interactivechaos.com/es/manual/tutorial-de-machine-learning/adam>