Anexo 11

Proyecto 11: Detector de Spam

Mg. Luis Felipe Bustamante Narváez

Este proyecto se desarrollará utilizando una base de datos sintética creada a través de ChatGPT con base en datasets extraídos en inglés de diferentes repositorios. Esto debido a, no tener muchas fuentes de gran tamaño de correos electrónicos en español, y no poder hacer uso de ciertas bases de datos por motivos de seguridad. Además, se utilizará una base de datos tomada de kaggle, donde se muestran correos en inglés, para validar que sin importar el idioma, se cumple con el objetivo.

Para desarrollar este ejercicio, usaremos Naive Bayes, con el fin de clasificar si un correo electrónico es o no, spam.

Librerías

In [...]
```
pip install seaborn -q
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [...]
```
pip install wordcloud -q
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [...]
```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
from sklearn.naive_bayes import MultinomialNB
from wordcloud import WordCloud  #Mostrar gráfico de palabras
```

Cargamos los datos

In [...]
```
opcion = int(input('Ingrese 1 para los correos en español y 2 para los correos en inglés:'))

if opcion == 1:
    path = 'data/spam.csv'
    df = pd.read_csv(path, encoding='utf-8')
elif opcion == 2:
    path = 'data/spam_or_not_spam.csv'
    df = pd.read_csv(path, encoding='ISO-8859-1')
    df.rename(columns={'email':'contenido', 'label':'spam'}, inplace=True)
    #encontramos una cadena NAN  df[df.isna().any(axis=1)]
    df['contenido'] = df['contenido'].fillna('')  #La reemplazamos por cadena vacía
else:
    print('Opción no válida')
```

In [...]
```
df
```

Out[…

	contenido	spam
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0
1	martin a posted tassos papadopoulos the greek ...	0
2	man threatens explosion in moscow thursday aug...	0
3	klez the virus that won t die already the most...	0
4	in adding cream to spaghetti carbonara which ...	0
...	...	...
2995	abc s good morning america ranks it the NUMBE...	1
2996	hyperlink hyperlink hyperlink let mortgage le...	1
2997	thank you for shopping with us gifts for all ...	1
2998	the famous ebay marketing e course learn to s...	1
2999	hello this is chinese traditional å□ ä»¶ NUM...	1

3000 rows × 2 columns

In […

```
df['contenido'][0][:500]
```

Out[…

```
' date wed NUMBER aug NUMBER NUMBER NUMBER NUMBER NUMBER from chris garrigues cwg dated NUMB
ER NUMBERfaNUMBERd deepeddy com message id NUMBER NUMBER tmda deepeddy vircio com i can t re
produce this error for me it is very repeatable like every time without fail this is the deb
ug log of the pick happening NUMBER NUMBER NUMBER pick_it exec pick inbox list lbrace lbrace
subject ftp rbrace rbrace NUMBER NUMBER sequence mercury NUMBER NUMBER NUMBER exec pick inbo
x list lbrace lbrace subject ftp rbrace '
```

In […

```
df['spam'][0]
```

Out[…

```
0
```

In […

```
# Agrupamos para obtener el total de datos  (0-> ham, 1->spam)
grouped = df.groupby('spam').count()
grouped
```

Out[…

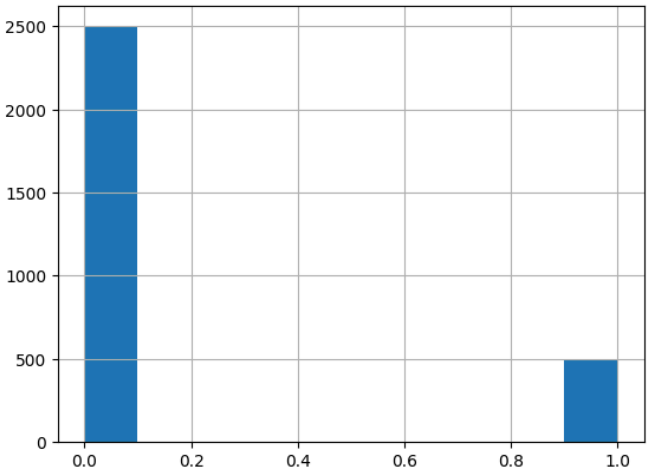
spam	contenido
0	2500
1	500

In […

```
# Gráfico de histograma de la población
df['spam'].hist()
```

Out[…

```
<Axes: >
```



In […

```
# Convertimos la columna spam en un arreglo
Y = df['spam'].to_numpy()
```

In [...]
```
Y
```

Out[...]
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

Procesamiento de Datos

Entrenamiento

In [...]
```
# Dividimos los datos en conjuntos de entrenamiento y de prueba
df_train, df_test, Y_train, Y_test = train_test_split(df['contenido'], Y, test_size=0.2)
```

In [...]
```
df_train
```

Out[...]
```
244      zimbabwe has dropped objections to accepting ...
1157    on wed aug NUMBER NUMBER at NUMBER NUMBER ulis...
2418    url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...
171     david asked my wife noticed something odd the ...
1458    on wed NUMBER sep NUMBER stephane lentz wrote ...
                              ...
294      URL an investigation has been launched after ...
1644     your mail and gives you only the non spam to ...
2648     unlimited web conferencing subscribe to the w...
2599     free adult lifetime membership limited time o...
948      from valdis kletnieks URL date mon NUMBER aug...
Name: contenido, Length: 2400, dtype: object
```

In [...]
```
df_test
```

Out[...]
```
1585    i m taking all my razored mail today and calli...
1495     URL jm URL changed what removed added status ...
855      original message from gary lawrence murphy ga...
1171    help i had gpg working i updated from version ...
2227    url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...
                              ...
675      help me out here you around barely but don t ...
90      hi dermot if have a look at one of the dists l...
197     hey i has just been given an old toshiba csNUM...
1506    unable to find user matt_relay sbcglobal net p...
354     on wed NUMBER NUMBER NUMBER at NUMBER NUMBER g...
Name: contenido, Length: 600, dtype: object
```

In [...]
```
len(Y_train)
```

Out[...]
```
2400
```

Vectorizamos

In [...]
```
vectores = CountVectorizer(decode_error='ignore')
X_train = vectores.fit_transform(df_train)
X_test = vectores.transform(df_test)
```

In [...]
```
X_train
```

Out[...]
```
<2400x31260 sparse matrix of type '<class 'numpy.int64'>'
        with 284220 stored elements in Compressed Sparse Row format>
```

In [...]
```
X_test
```

Out[...]
```
<600x31260 sparse matrix of type '<class 'numpy.int64'>'
        with 61327 stored elements in Compressed Sparse Row format>
```

Modelo

In [...
```
model = MultinomialNB()
model.fit(X_train, Y_train)
```

Out[...
```
MultinomialNB()
```

In [...
```
# Probamos el modelo con los datos originales
train_accuracy = model.score(X_train, Y_train)
test_accuracy = model.score(X_test, Y_test)
```

In [...
```
# Mostramos la puntuación
print(f'El accuracy de entrenamiento es de {train_accuracy}')
print(f'El accuracy de prueba es de {test_accuracy}')
```

```
El accuracy de entrenamiento es de 0.9954166666666666
El accuracy de prueba es de 0.9983333333333333
```

In [...
```
# Probamos la predicción del modelo
P_train = model.predict(X_train)
P_test = model.predict(X_test)
```

In [...
```
# Mostramos el ajuste del modelo predicho con f1
print(f'Train F1: {f1_score(Y_train, P_train)}')
print(f'Test F1: {f1_score(Y_test, P_test)}')
```

```
Train F1: 0.9864029666254636
Test F1: 0.994535519125683
```

Matriz de Confusión

In [...
```
conf_matrix_train = confusion_matrix(Y_train, P_train)
conf_matrix_train
```

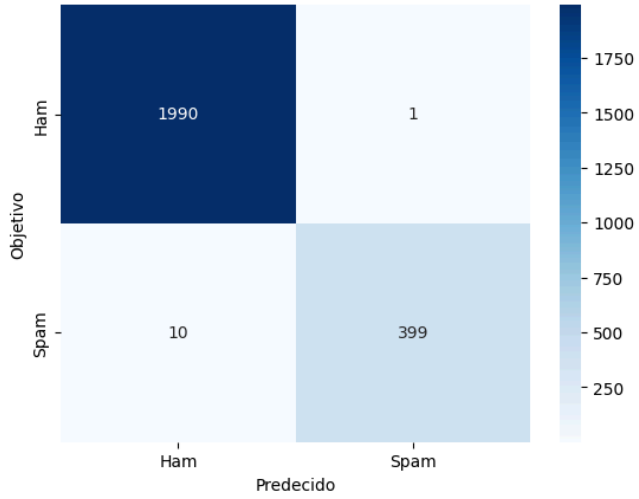
Out[...
```
array([[1990,    1],
       [  10,  399]], dtype=int64)
```

In [...
```
conf_matrix_test = confusion_matrix(Y_test, P_test)
conf_matrix_test
```

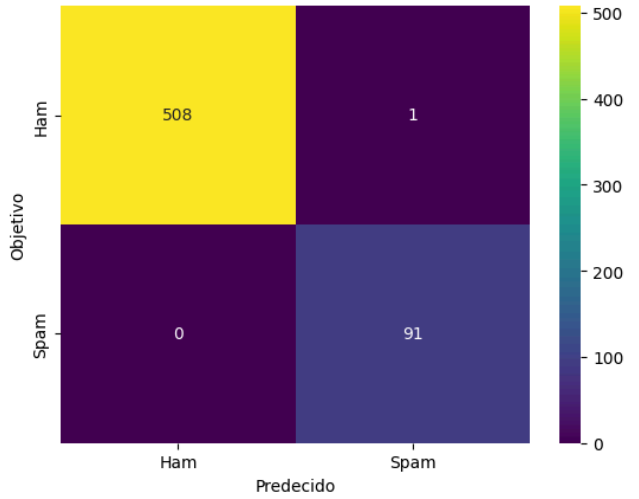
Out[...
```
array([[508,   1],
       [  0,  91]], dtype=int64)
```

In [...
```
# Gráfico de la matriz de confusión
def plot_conf_matrix(c_m, color):
    classes = ['Ham', 'Spam']
    df_cm = pd.DataFrame(c_m, index=classes, columns=classes)
    ax = sn.heatmap(df_cm, annot=True, fmt='g', cmap=color)
    ax.set_xlabel('Predecido')
    ax.set_ylabel('Objetivo')
```

In [...
```
color = 'Blues'  #coolwarm / viridis / Blues / Greens / Reds / magma / cividis
plot_conf_matrix(conf_matrix_train, color)
```



In [...]
```
color = 'viridis'
plot_conf_matrix(conf_matrix_test, color)
```

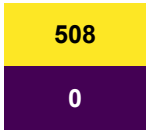


La matriz de correlación, permite visualizar la cantidad de correos que se analizaron en el grupo de entrenamiento y de prueba, y se puede interpretar de la siguiente manera:

- La fila **0**, habla de los correos esperados que **spam**.

- La fila **1**, habla de los correos esperados que son **spam**.

- La columna **0**, habla de los correos predichos que **no spam**.



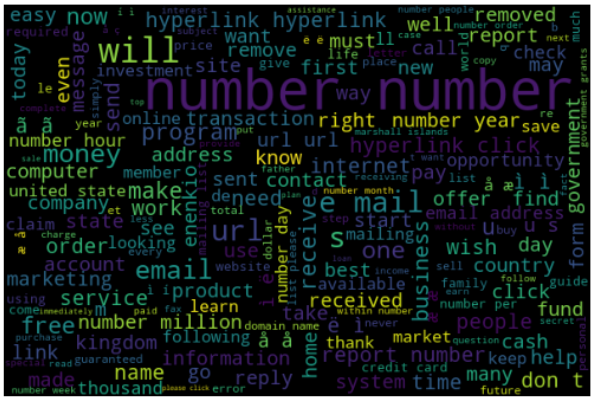
- La columna **1**, habla de los correos predichos que son **spam**.

- La celda **(0, 0)**, muestra los correos que **no spam**, que se esperaban y que se predijeron correctamente.
- La celda **(1, 1)**, muestra los correos que son **spam**, que se esperaban y que también se predijeron correctamente.
- La celda **(0, 1)**, muestra los correos que se esperaban como **spam** y que la predicción los arrojó como **no spam**, es decir, los correos tipo **spam** que se lograron colar como correos buenos.
- La celda **(1, 0)**, muestra los correos que se esperaban como **no spam** y que la predicción los arrojó como **spam**, es decir, los falsos **spam** positivos.

WordCloud

In [...
```
def visualize(label):
    words = ''
    for msg in df[df['spam'] == label]['contenido']:
        msg = msg.lower()
        words += msg + ' '
    wordcloud = WordCloud(width=600, height=400).generate(words)
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()
```

In [...
```
visualize(1)  #enviamos un 1, ya que la columna spam es 1 si el correo es spam
```



Explicación

Este generador de nube de palabras, se crea a partir de una cadena **words** vacía, la cual se va llenando cada vez que al recorrer el ciclo **for**, se notan con mayor frecuencia ciertas palabras en los mensaje clasificados como **spam**. Se convierten las palabras a minúsculas y se muestran utilizando el método **WordCloud** de la librería que lleva su mismo nombre.

Entre más se repite una palabra, más grande se ve en la nube de palabras.

Identificación de Falsos Spam

In [...
```
# Vectorizamos la columna contenido
X = vectores.transform(df['contenido'])
# Cremos la columna de predicciones
df['predicciones'] = model.predict(X)
df
```

Out[…

	contenido	spam	predicciones
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0	0
1	martin a posted tassos papadopoulos the greek ...	0	0
2	man threatens explosion in moscow thursday aug...	0	0
3	klez the virus that won t die already the most...	0	0
4	in adding cream to spaghetti carbonara which ...	0	0
...	...	...	...
2995	abc s good morning america ranks it the NUMBE...	1	1
2996	hyperlink hyperlink hyperlink let mortgage le...	1	1
2997	thank you for shopping with us gifts for all ...	1	1
2998	the famous ebay marketing e course learn to s...	1	1
2999	hello this is chinese traditional å□ ä»¶ NUM...	1	1

3000 rows × 3 columns

In […

```python
# identificación de los falsos positivos
falso_spam = df[(df['predicciones'] == 1) & (df['spam'] == 0)]['contenido']
falso_ham = df[(df['predicciones'] == 0) & (df['spam'] == 1)]['contenido']
if falso_spam.empty:
    print('No se encontraron falsos Spam')
else:
    print('**Falsos Spam**\n')
    for msg in falso_spam:
        print(msg[:300])

if falso_ham.empty:
    print('\n\nNo se encontraron falsos Ham')
else:
    print('\n\n**Falsos Ham**\n')
    for msg in falso_ham:
        print(msg[:10])
```

```
**Falsos Spam**

 with our telecoms partner bumblebee don t get ripped off by expensive hotel payphone and mob
ile charges save save save on international calls with ryanair s phone partner you ll save up
to NUMBER on international phone calls when you use our online phone card you can use the car
d from any phone in
url URL date not supplied detailed guidelines for vaccinating all NUMBER million citizens wit
hin five days of an outbreak are being dispatched to every state


**Falsos Ham**

 NUMBER NU
 x m a h c
 this URL
 r v r f i

 r v r f i
 r v r f i

 this URL
```

Conclusiones

Se realizó un modelo basado en datos de correos electrónicos en español e inglés, obteniendo resultados diferentes pero con alta probabilidad de clasificación. Esto permite identificar que los modelos de Naive Bayes, en este caso el Multinomial, permiten realizar un óptimo proceso para separar correos basura.

Mg. Luis Felipe Bustamante Narváez

Anexo 12

Proyecto 12: Analisis de Sentimiento

Mg. Luis Felipe Bustamante Narváez

En este proyecto vamos a analizar unos dataset que contienen opiniones de usuarios y un análisis previo sobre dichas opiniones, si estas son positivas, negativas o neutras. El objetivo es predecir emociones o sentimientos de las personas, basado en los comentarios que dejan al respecto de un producto, situación o cualquier variable que apele a un calificativo.

Librerías

In [...]
```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
import itertools
```

Cargamos los datos

In [...]
```
path = 'data/comentarios_peliculas.csv'
df = pd.read_csv(path, encoding='utf-8')
```

In [...]
```
df
```

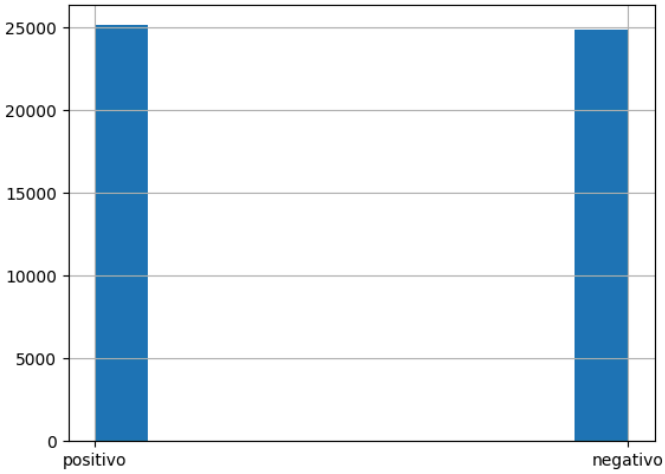
Out[...]

	comentario	sentimiento
0	Cada escena estaba llena de emoción y signific...	positivo
1	El guion es predecible y los actores no transm...	negativo
2	Los efectos especiales eran ridículos y poco c...	negativo
3	Muy mala, esperaba mucho más y me decepcionó.	negativo
4	Una obra maestra, la mejor película que he vis...	positivo
...	...	...
49995	Cada escena estaba llena de emoción y signific...	positivo
49996	El guion es predecible y los actores no transm...	negativo
49997	Una película increíble, me encantó cada escena.	positivo
49998	Terrible, no entiendo cómo alguien puede disfr...	negativo
49999	La historia es fascinante y los actores hicier...	positivo

50000 rows × 2 columns

In [...]
```
# histograma de sentimientos
df['sentimiento'].hist()
```

Out[...] <Axes: >



In [...
```
# Anexamos una columna binaria (objetivo)
target_map = {'positivo': 1, 'negativo': 0}
df['target'] = df['sentimiento'].map(target_map)
df
```

Out[...

	comentario	sentimiento	target
0	Cada escena estaba llena de emoción y signific...	positivo	1
1	El guion es predecible y los actores no transm...	negativo	0
2	Los efectos especiales eran ridículos y poco c...	negativo	0
3	Muy mala, esperaba mucho más y me decepcionó.	negativo	0
4	Una obra maestra, la mejor película que he vis...	positivo	1
...	...	...	...
49995	Cada escena estaba llena de emoción y signific...	positivo	1
49996	El guion es predecible y los actores no transm...	negativo	0
49997	Una película increíble, me encantó cada escena.	positivo	1
49998	Terrible, no entiendo cómo alguien puede disfr...	negativo	0
49999	La historia es fascinante y los actores hicier...	positivo	1

50000 rows × 3 columns

In [...
```
# Agrupamos para obtener el total de datos  (0-> negativo, 1->positivo)
grouped = df.groupby('target').count()
grouped
```

Out[...

target	comentario	sentimiento
0	24871	24871
1	25129	25129

Procesamiento de los Datos

Entrenamiento

In [...
```
df_train, df_test = train_test_split(df)
```

In [...
```
df_train
```

Out[…

	comentario	sentimiento	target
34636	Me hizo reír, llorar y reflexionar, todo en un...	positivo	1
34951	Los efectos especiales eran ridículos y poco c...	positivo	1
27374	El desarrollo de los personajes es pobre y sin...	negativo	0
32526	El final fue absurdo y dejó muchas preguntas s...	positivo	1
45042	Una película increíble, me encantó cada escena.	positivo	1
...	...	...	...
10413	El final fue absurdo y dejó muchas preguntas s...	negativo	0
9468	El guion es predecible y los actores no transm...	negativo	0
11220	El desarrollo de los personajes es pobre y sin...	negativo	0
9298	No tiene ni pies ni cabeza, simplemente mala.	negativo	0
35665	El desarrollo de los personajes es pobre y sin...	negativo	0

37500 rows × 3 columns

In […

```
df_test
```

Out[…

	comentario	sentimiento	target
22896	El final fue absurdo y dejó muchas preguntas s...	positivo	1
45709	Muy entretenida, sin duda la volvería a ver.	positivo	1
12283	Una pérdida de tiempo total, me arrepiento de ...	negativo	0
43490	Muy mala, esperaba mucho más y me decepcionó.	negativo	0
4261	Me aburrí desde el primer minuto, no la recomi...	negativo	0
...	...	...	...
42442	Muy entretenida, sin duda la volvería a ver.	positivo	1
27913	El desarrollo de los personajes es pobre y sin...	negativo	0
38611	El guion es predecible y los actores no transm...	negativo	0
7900	Una película increíble, me encantó cada escena.	positivo	1
14261	El guion es predecible y los actores no transm...	negativo	0

12500 rows × 3 columns

Vectorización

In […

```
# usamos un máximo de 2000 dimensiones
vectorizer = TfidfVectorizer(max_features=2000)
```

In […

```
# vectorizamos el entrenamiento
X_train = vectorizer.fit_transform(df_train['comentario'])
X_train
```

Out[…

```
<37500x109 sparse matrix of type '<class 'numpy.float64'>'
        with 305339 stored elements in Compressed Sparse Row format>
```

In […

```
X_test = vectorizer.transform(df_test['comentario'])
X_test
```

Out[…

```
<12500x109 sparse matrix of type '<class 'numpy.float64'>'
        with 101755 stored elements in Compressed Sparse Row format>
```

In […

```
Y_train = df_train['target']
Y_test = df_test['target']
```

In […

```
len(Y_train)
```

Out[…

```
37500
```

In […

```
len(Y_test)
```

Out[… 12500

Modelo

In […
```
model = LogisticRegression(max_iter=500)  #Genera 500 iteraciones cambiando los pesos/sesgos
model.fit(X_train, Y_train)
```

Out[…
```
LogisticRegression
LogisticRegression(max_iter=500)
```

In […
```
# Probamos el modelo con los datos originales
train_accuracy = model.score(X_train, Y_train)
test_accuracy = model.score(X_test, Y_test)
```

In […
```
# Mostramos la puntuación
print(f'El accuracy de entrenamiento es de {train_accuracy}')
print(f'El accuracy de prueba es de {test_accuracy}')
```

```
El accuracy de entrenamiento es de 0.95088
El accuracy de prueba es de 0.94736
```

Predicciones

In […
```
P_train = model.predict(X_train)
P_test = model.predict(X_test)
```

In […
```
# Matriz de confusión
conf_matrix_train = confusion_matrix(Y_train, P_train, normalize='true')
conf_matrix_train
```

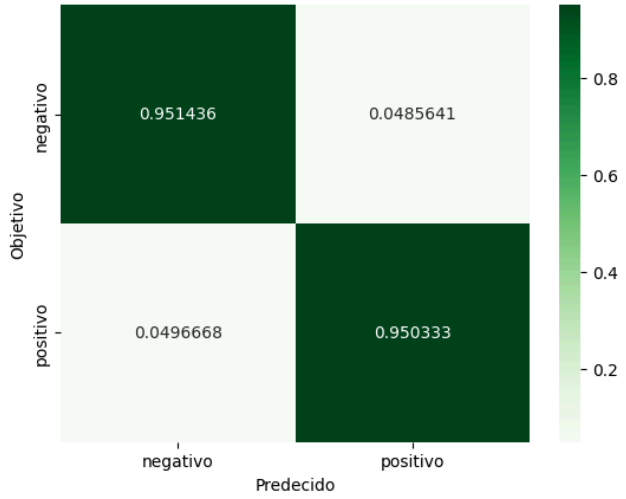
Out[…
```
array([[0.95143595, 0.04856405],
       [0.04966677, 0.95033323]])
```

In […
```
conf_matrix_test = confusion_matrix(Y_test, P_test, normalize='true')
conf_matrix_test
```

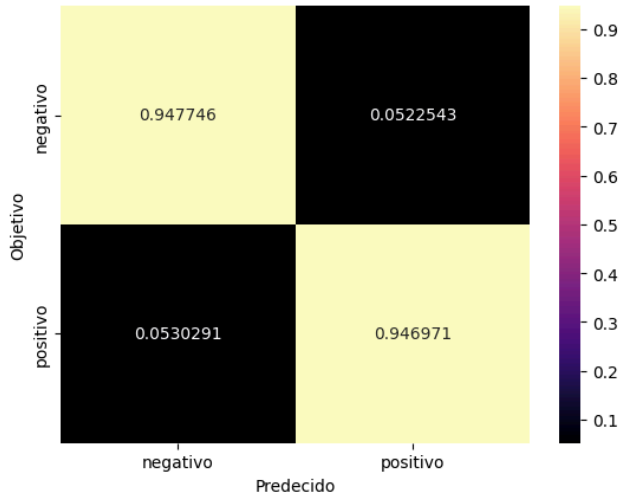
Out[…
```
array([[0.94774574, 0.05225426],
       [0.05302909, 0.94697091]])
```

In […
```
# Gráfico de la matriz de confusión
def plot_conf_matrix(c_m, color):
    classes = ['negativo', 'positivo']
    df_cm = pd.DataFrame(c_m, index=classes, columns=classes)
    ax = sn.heatmap(df_cm, annot=True, fmt='g', cmap=color)
    ax.set_xlabel('Predecido')
    ax.set_ylabel('Objetivo')
```

In […
```
color = 'Greens'  #coolwarm / viridis / Blues / Greens / Reds / magma / cividis
plot_conf_matrix(conf_matrix_train, color)
```



In [...
```
color = 'magma'
plot_conf_matrix(conf_matrix_test, color)
```



Análisis de las palabras

In [...
```
# vectorización de palabras
word_index_map = vectorizer.vocabulary_
dict(itertools.islice(word_index_map.items(), 5))
```

Out[...
```
{'me': 56, 'hizo': 45, 'reír': 87, 'llorar': 51, 'reflexionar': 85}
```

In [...
```
# Coeficiente de las palabras
model.coef_[0][:10]
```

Out[...
```
array([-1.44036264e+00, -1.15267244e+00,  4.48615605e-04,  6.22903789e-01,
       -1.09387316e+00, -1.21140622e+00, -8.04303645e-01,  5.58432574e-01,
       -7.37314123e-01,  2.10564178e+00])
```

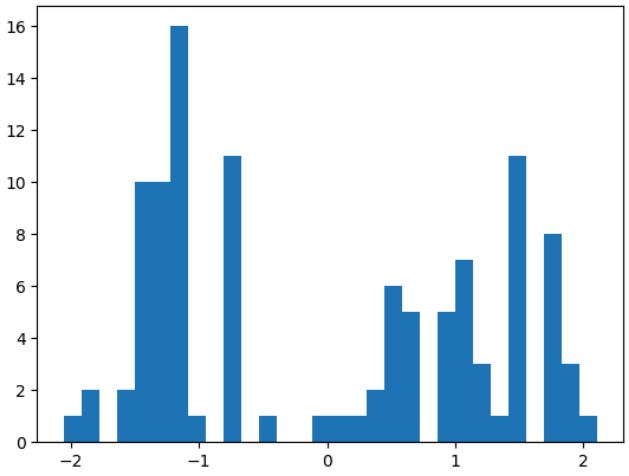
In [...
```
limit = 1.5
print('**Palabras más positivas**\n')
for word, index in word_index_map.items():
    weight = model.coef_[0][index]
    if weight > limit:
        print(word, weight)
```

```
**Palabras más positivas**

cada 2.1056417823800224
escena 1.8834508871239404
perfecto 1.7561162834981257
podría 1.7561162834981257
haber 1.7561162834981257
sido 1.7561162834981257
mejor 1.9163405384954952
la 1.7578370103429033
emoción 1.5203371110754182
excelente 1.7906494248075528
visuales 1.7906494248075528
impresionantes 1.7906494248075528
un 1.939566627288709
```

In [...
```
limit = 1.5
print('**Palabras más negativas**\n')
for word, index in word_index_map.items():
    weight = model.coef_[0][index]
    if weight < -limit:
        print(word, weight)
```

```
**Palabras más negativas**

el -1.7793619395843912
de -1.8047724408415342
no -1.571073260466105
mala -2.051899328241801
esta -1.5697751259113608
```

In [...
```
# Gráfico de pesos
plt.hist(model.coef_[0], bins=30)
plt.show()
```



Probamos el modelo

In [...
```
prueba = ['estuvo muy entretenida la película',
          'estuvo horrible la película, me aburrí mucho',
         'no la recomiendo']
```

In [...
```
# Vectorizamos la prueba
x = vectorizer.transform(prueba)
```

```
x
```

Out[…
```
<3x109 sparse matrix of type '<class 'numpy.float64'>'
        with 12 stored elements in Compressed Sparse Row format>
```

In […
```
# Predecimos con el modelo
P = model.predict(x)
```

In […
```
# Obtenemos las clases del modelo
clases = model.classes_
```

In […
```
# Mostramos la clase de la prueba
for i in range(len(prueba)):
    if clases[P[i]] == 0:
        print(f'El comentario "{prueba[i]}" es Negativo')
    else:
        print(f'El comentario "{prueba[i]}" es Positivo')
```

```
El comentario "estuvo muy entretenida la película" es Positivo
El comentario "estuvo horrible la película, me aburrí mucho" es Negativo
El comentario "no la recomiendo" es Negativo
```

Conclusiones

Se realizó un modelo basado en datos de comentarios sobre películas, obteniendo resultados diferentes pero con alta probabilidad de clasificación. Esto permite identificar que la función Sigmoide permiten realizar un óptimo proceso para separar reconocer sentimientos positivos y negativos.

Mg. Luis Felipe Bustamante Narváez

Anexo 13

Proyecto 13: Analisis de Sentimiento Multiclase

Mg. Luis Felipe Bustamante Narváez

En este proyecto vamos a analizar unos dataset que contienen opiniones de usuarios y un análisis previo sobre dichas opiniones, si estas son positivas, negativas o neutras. El objetivo es predecir emociones o sentimientos de las personas, basado en los comentarios que dejan en redes sociales, para el ejemplo, facebook.

A diferencia del Proyecto 12, en este caso, usaremos multiclases, de tal manera que se puedan analizar más de dos sentimientos, y el proceso deje de ser binario.

Librerías

In [...]
```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
import itertools
```

Cargamos los datos

In [...]
```
path = 'data/comentarios_facebook.csv'
df = pd.read_csv(path, encoding='utf-8')
```

In [...]
```
df
```

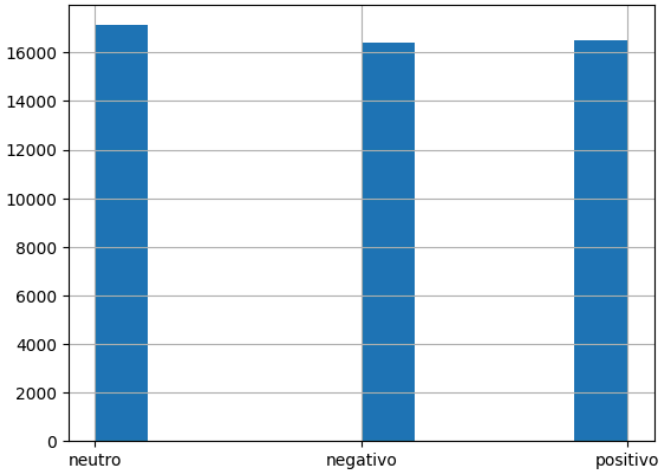
Out[...]

	comentario	sentimiento
0	No afecta mi vida, pero es interesante saberlo.	neutro
1	Esto es una completa pérdida de tiempo.	negativo
2	Buenísima recomendación, la probaré!	positivo
3	Es un buen punto, aunque depende de la perspec...	neutro
4	Qué publicación tan aburrida y sin sentido.	negativo
...	...	...
49995	Me encanta esta publicación, muy inspiradora!	neutro
49996	Me decepcionó mucho este contenido.	negativo
49997	Ni bueno ni malo, simplemente un comentario más.	neutro
49998	No entiendo por qué esto está en mi feed.	negativo
49999	No estoy seguro de qué pensar al respecto.	neutro

50000 rows × 2 columns

In [...]
```
# histograma de sentimientos
df['sentimiento'].hist()
```

Out[...] <Axes: >



In [...
```
# Anexamos una columna binaria (objetivo)
target_map = {'positivo': 1, 'negativo': 0, 'neutro': 2}
df['target'] = df['sentimiento'].map(target_map)
df
```

Out[...

	comentario	sentimiento	target
0	No afecta mi vida, pero es interesante saberlo.	neutro	2
1	Esto es una completa pérdida de tiempo.	negativo	0
2	Buenísima recomendación, la probaré!	positivo	1
3	Es un buen punto, aunque depende de la perspec...	neutro	2
4	Qué publicación tan aburrida y sin sentido.	negativo	0
...	...	...	...
49995	Me encanta esta publicación, muy inspiradora!	neutro	2
49996	Me decepcionó mucho este contenido.	negativo	0
49997	Ni bueno ni malo, simplemente un comentario más.	neutro	2
49998	No entiendo por qué esto está en mi feed.	negativo	0
49999	No estoy seguro de qué pensar al respecto.	neutro	2

50000 rows × 3 columns

In [...
```
# Agrupamos para obtener el total de datos  (0-> negativo, 1->positivo)
grouped = df.groupby('target').count()
grouped
```

Out[...

target	comentario	sentimiento
0	16384	16384
1	16495	16495
2	17121	17121

Procesamiento de los Datos

Entrenamiento

In [...
```
df_train, df_test = train_test_split(df)
```

In [...
```
df_train
```

Out[…

	comentario	sentimiento	target
46633	Interesante, pero me gustaría más información.	neutro	2
14472	Gran trabajo, sigue así!	positivo	1
19520	Ni bueno ni malo, simplemente un comentario más.	neutro	2
1674	Podría ser cierto, aunque hay otras opiniones.	neutro	2
27658	Es un buen punto, aunque depende de la perspec...	positivo	1
...	...	...	...
6088	Algunas partes son ciertas, otras no tanto.	neutro	2
24509	Me hiciste reír, gracias por compartir!	positivo	1
32483	Me hiciste reír, gracias por compartir!	positivo	1
42789	Me decepcionó mucho este contenido.	negativo	0
43208	Es un buen punto, aunque depende de la perspec...	neutro	2

37500 rows × 3 columns

In […

```
df_test
```

Out[…

	comentario	sentimiento	target
1745	Tu contenido ha ido empeorando cada vez más.	negativo	0
5583	Demasiado exagerado, no vale la pena leerlo.	negativo	0
47317	Tu contenido ha ido empeorando cada vez más.	negativo	0
457	Malísima recomendación, no la sigan.	negativo	0
31703	Me decepcionó mucho este contenido.	negativo	0
...	...	...	...
8044	No estoy de acuerdo para nada, pésimo argumento.	negativo	0
21421	No entiendo por qué esto está en mi feed.	negativo	0
38151	Ni bueno ni malo, simplemente un comentario más.	neutro	2
4594	No tiene sentido lo que dices, muy decepcionante.	negativo	0
4596	Me decepcionó mucho este contenido.	negativo	0

12500 rows × 3 columns

Vectorización

In […

```
# usamos un máximo de 2000 dimensiones
vectorizer = TfidfVectorizer(max_features=2000)
```

In […

```
# vectorizamos el entrenamiento
X_train = vectorizer.fit_transform(df_train['comentario'])
X_train
```

Out[…

```
<37500x132 sparse matrix of type '<class 'numpy.float64'>'
        with 256951 stored elements in Compressed Sparse Row format>
```

In […

```
X_test = vectorizer.transform(df_test['comentario'])
X_test
```

Out[…

```
<12500x132 sparse matrix of type '<class 'numpy.float64'>'
        with 85272 stored elements in Compressed Sparse Row format>
```

In […

```
Y_train = df_train['target']
Y_test = df_test['target']
```

In […

```
len(Y_train)
```

Out[…

```
37500
```

In […

```
len(Y_test)
```

Out[… 12500

Modelo

In […
```
model = LogisticRegression(max_iter=1000)  #Genera 1000 iteraciones cambiando los pesos/sesgo
model.fit(X_train, Y_train)
```

Out[…
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In […
```
# Probamos el modelo con los datos originales
train_accuracy = model.score(X_train, Y_train)
test_accuracy = model.score(X_test, Y_test)
```

In […
```
# Mostramos la puntuación
print(f'El accuracy de entrenamiento es de {train_accuracy}')
print(f'El accuracy de prueba es de {test_accuracy}')
```

```
El accuracy de entrenamiento es de 0.9496533333333333
El accuracy de prueba es de 0.95104
```

Predicciones

In […
```
P_train = model.predict(X_train)
P_test = model.predict(X_test)
```

In […
```
# Matriz de confusión
conf_matrix_train = confusion_matrix(Y_train, P_train, normalize='true')
conf_matrix_train
```

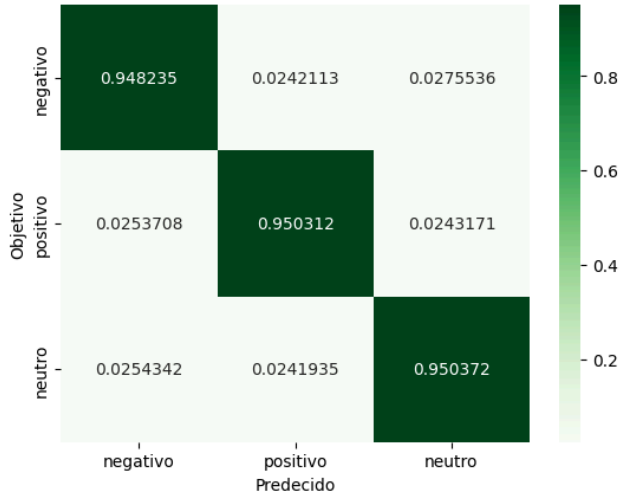
Out[…
```
array([[0.9482351 , 0.0242113 , 0.0275536 ],
       [0.02537084, 0.95031207, 0.02431709],
       [0.02543424, 0.02419355, 0.95037221]])
```

In […
```
conf_matrix_test = confusion_matrix(Y_test, P_test, normalize='true' )
conf_matrix_test
```

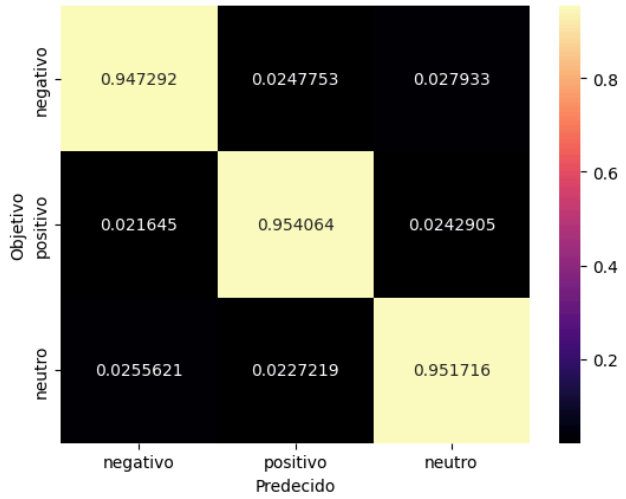
Out[…
```
array([[0.94729172, 0.02477532, 0.02793296],
       [0.02164502, 0.95406445, 0.02429052],
       [0.02556213, 0.02272189, 0.95171598]])
```

In […
```
# Gráfico de la matriz de confusión
def plot_conf_matrix(c_m, color):
    classes = ['negativo', 'positivo', 'neutro']
    df_cm = pd.DataFrame(c_m, index=classes, columns=classes)
    ax = sn.heatmap(df_cm, annot=True, fmt='g', cmap=color)
    ax.set_xlabel('Predecido')
    ax.set_ylabel('Objetivo')
```

In […
```
color = 'Greens'  #coolwarm / viridis / Blues / Greens / Reds / magma / cividis
plot_conf_matrix(conf_matrix_train, color)
```



In [...
```
color = 'magma'
plot_conf_matrix(conf_matrix_test, color)
```



Análisis de las palabras

In [...
```
# vectorización de palabras
word_index_map = vectorizer.vocabulary_
dict(itertools.islice(word_index_map.items(), 5))
```

Out[...
```
{'interesante': 61, 'pero': 87, 'me': 69, 'gustaría': 51, 'más': 76}
```

In [...
```
# Coeficiente de las palabras
model.coef_[0].max()
```

Out[...
```
2.1007847353587525
```

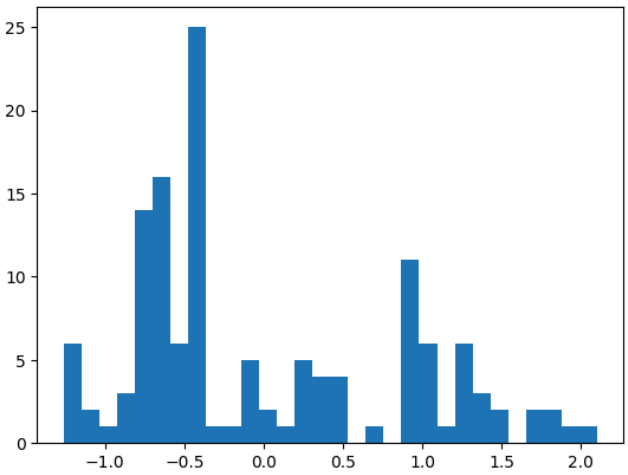
In [...
```
limit = 1.5
print('**Palabras más negativas**\n')
for word, index in word_index_map.items():
    weight = model.coef_[0][index]
    if weight > limit:
        print(word, weight)
```

```
**Palabras más negativas**

no 1.5109386310454178
esto 2.1007847353587525
decepcionó 1.8484200261758081
este 1.8484200261758081
sentido 1.9737735487852108
malísima 1.6576414288486296
sigan 1.6576414288486296
```

In [...
```
limit = 1.1
print('**Palabras más positivas**\n')
for word, index in word_index_map.items():
    weight = model.coef_[0][index]
    if weight < -limit:
        print(word, weight)
```

```
**Palabras más positivas**

pero -1.1952465233388108
buenísima -1.2596535749701068
probaré -1.2596535749701068
excelente -1.1692834161609396
al -1.1574146905115645
respecto -1.1574146905115645
```

In [...
```
# Gráfico de pesos
plt.hist(model.coef_[0], bins=30)
plt.show()
```



Probamos el modelo

In [...
```
prueba = ['estuvo muy entretenida la película',
          'estuvo horrible la película, me aburrí mucho',
         'no la recomiendo']
```

In [...
```
# Vectorizamos la prueba
x = vectorizer.transform(prueba)
x
```

Out[...
```
<3x132 sparse matrix of type '<class 'numpy.float64'>'
        with 7 stored elements in Compressed Sparse Row format>
```

In [...
```python
# Predecimos con el modelo
P = model.predict(x)
```

In [...
```python
# Obtenemos las clases del modelo
clases = model.classes_
```

In [...
```python
# Mostramos la clase de la prueba
for i in range(len(prueba)):
    if clases[P[i]] == 0:
        print(f'El comentario "{prueba[i]}" es Negativo')
    elif clases[P[i]] == 2:
        print(f'El comentario "{prueba[i]}" es Neutro')
    else:
        print(f'El comentario "{prueba[i]}" es Positivo')
```

```
El comentario "estuvo muy entretenida la película" es Positivo
El comentario "estuvo horrible la película, me aburrí mucho" es Positivo
El comentario "no la recomiendo" es Negativo
```

Conclusiones

Se realizó un modelo basado en datos de comentarios en redes sociales, obteniendo resultados diferentes pero con alta probabilidad de clasificación. Esto permite identificar que la función Softmax permiten realizar un óptimo proceso para separar y reconocer sentimientos positivos y negativos y de otras características..

Mg. Luis Felipe Bustamante Narváez

Anexo 14

Proyecto 14: Resumen de textos con Vectorización

Mg. Luis Felipe Bustamante Narváez

En este proyecto, desarrollaremos un generador de resúmenes a través de vectorización, tomando una base de datos importante, de diferentes artículos de prensa para analizar cómo pueden sumarizarse y permitir los beneficios de la inteligencia artificial.

Librerías

In [...
```
import pandas as pd
import numpy as np
import textwrap
import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [...
```
nltk.download('punkt')
nltk.download('stopwords')
```

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

Out[... True

In [...
```
path = 'data/df_total.csv'
df = pd.read_csv(path, encoding='utf-8')
```

In [...
```
df
```

Out[...

	url	news	Type
0	https://www.larepublica.co/redirect/post/3201905	Durante el foro La banca articulador empresari...	Otra
1	https://www.larepublica.co/redirect/post/3210288	El regulador de valores de China dijo el domin...	Regulaciones
2	https://www.larepublica.co/redirect/post/3240676	En una industria históricamente masculina como...	Alianzas
3	https://www.larepublica.co/redirect/post/3342889	Con el dato de marzo el IPC interanual encaden...	Macroeconomia
4	https://www.larepublica.co/redirect/post/3427208	Ayer en Cartagena se dio inicio a la versión n...	Otra
...	...	...	...
1212	https://www.bbva.com/es/como-lograr-que-los-in...	En la vida de toda empresa emergente llega un ...	Innovacion
1213	https://www.bbva.com/es/podcast-como-nos-afect...	La espiral alcista de los precios continúa y g...	Macroeconomia
1214	https://www.larepublica.co/redirect/post/3253735	Las grandes derrotas nacionales son experienci...	Alianzas
1215	https://www.bbva.com/es/bbva-y-barcelona-healt...	BBVA ha alcanzado un acuerdo de colaboración c...	Innovacion
1216	https://www.larepublica.co/redirect/post/3263980	Casi entrando a la parte final de noviembre la...	Alianzas

1217 rows × 3 columns

In [...
```
print(df['news'][2][:500])
```

```
En una industria históricamente masculina como lo es la aviación Viva presentó su avión rosad
o A320NEO que apuesta por la equidad de género la lucha contra el cáncer de mama la inclusión
y la diversidad.Desde Francia llegó Go Pink que tuvo un precio promedio de US50 millones conv
irtiéndose en la aeronave número 20 de las 21 con las que finalizará el año esta aerolínea.En
Viva estamos trabajando muy fuerte para que haya más mujeres. Actualmente el grupo ejecutivo
está compuesto por 42 mujeres per
```

In [...
```
# Buscamos una noticia larga para tomar como ejemplo a la hora de hacer el resumen
doc = df['news'].sample()
```

In [...
```
print(doc.iloc[0][:500])
```

```
El actual brote de inflación es un momento de déjà vu para las personas que vivieron las subi
das de precios de principios de la década de 1980.La inflación de EE.UU. se aceleró a una tas
a anual de 75 en enero alcanzando un máximo de cuatro décadas. El índice de precios al consum
idor que mide lo que la gente paga por bienes y servicios estuvo el mes pasado en su nivel má
s alto desde febrero de 1982 en comparación con enero de hace un año según el Departamento de
Trabajo.Blaise Jones recuerda a su
```

In [...
```
# Obtener el índice de la noticia para manipulación de datos
indice = df.index[df['news'] == doc.iloc[0]].tolist()
print(indice)
```

```
[297]
```

In [...
```
# hacemos la prueba
print(df['news'][297][:100])
```

```
El actual brote de inflación es un momento de déjà vu para las personas que vivieron las subi
das de
```

Procesamiento de Datos

TextWrap

In [...
```
# Eliminamos las palabras cortadas de las líneas
doc2 = textwrap.fill(doc.iloc[0], replace_whitespace=False, fix_sentence_endings=True)
```

In [...
```
print(doc2[:500])
```

```
El actual brote de inflación es un momento de déjà vu para las
personas que vivieron las subidas de precios de principios de la
década de 1980.La inflación de EE.UU. se aceleró a una tasa anual de
75 en enero alcanzando un máximo de cuatro décadas.  El índice de
precios al consumidor que mide lo que la gente paga por bienes y
servicios estuvo el mes pasado en su nivel más alto desde febrero de
1982 en comparación con enero de hace un año según el Departamento de
Trabajo.Blaise Jones recuerda a s
```

Separación de líneas

In [...
```
# Podemos separar por líneas, por puntos o comas, la idea es conservar oraciones
# con ideas claras.
lineas = doc2.split('. ')  #Usamos punto espacio, por las siglas o números que pueden haber
lineas[:3]
```

Out[…
```
['El actual brote de inflación es un momento de déjà vu para las\npersonas que vivieron las subidas de precios de principios de la\ndécada de 1980.La inflación de EE.UU',
 'se aceleró a una tasa anual de\n75 en enero alcanzando un máximo de cuatro décadas',
 ' El índice de\nprecios al consumidor que mide lo que la gente paga por bienes y\nservicios estuvo el mes pasado en su nivel más alto desde febrero de\n1982 en comparación con enero de hace un año según el Departamento de\nTrabajo.Blaise Jones recuerda a su madre hablando sobre el aumento del\nprecio de la leche y la determinación de su padre de mantener baja la\nfactura de calefacción de su hogar tácticas que incluían bajar el\ntermostato a 62 grados a la hora de acostarse.Juro que podía ver mi\nrespiración cuando me levantaba dijo el doctor Jones ahora de 59 años\ny neurorradiólogo pediátrico en Cincinnati']
```

Tokenización

In […
```
# Creamos la tokenización usando las stopwords descargadas
tokenizar = TfidfVectorizer(stop_words=stopwords.words('spanish'), norm='l1')
```

In […
```
# Creamos la matriz
X = tokenizar.fit_transform(lineas)
X
```

Out[…
```
<26x433 sparse matrix of type '<class 'numpy.float64'>'
        with 560 stored elements in Compressed Sparse Row format>
```

In […
```
# Mostramos la matriz
filas, columnas = X.shape

for i in range(10):  #aquí ponemos las filas, pero al ser muchas el resultado es extenso.
    for j in range(10):
        print(X[i, j], end=' ') # imprime el elemento y un espacio en blanco
    print()  #deja el renglón

# Cada fila va a representar una palabra
# Cada columna va a representar cada una de las oraciones
```

```
0.0 0.0 0.0 0.0 0.0 0.04756870737569473 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.018470715419332654 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.08059297416517343 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

In […
```
# Promediamos los puntajes de cada una de las oraciones
def obtener_score(tfidf_row):
    x = tfidf_row[tfidf_row != 0]  # Elimina las oraciones que no tienen puntuación
    return x.mean()
```

In […
```
# Creamos el vector de puntuación y lo llenamos
scores = np.zeros(len(lineas))
for i in range(len(lineas)):
    score = obtener_score(X[i,:])
    scores[i] = score
```

In […
```
scores
```

Out[…
```
array([0.06666667, 0.11111111, 0.01754386, 0.05      , 0.07692308,
       0.0625    , 1.        , 0.25      , 0.04761905, 0.03703704,
       0.04545455, 0.02439024, 0.03846154, 0.02083333, 0.06666667,
       0.03225806, 0.04545455, 0.0625    , 0.05263158, 0.03225806,
       0.125     , 0.09090909, 0.03703704, 0.07692308, 0.03030303,
       0.07142857])
```

Resumen

In […
```
# Ordenamos los scores y mostramos las posiciones de mayor a menor
sort_index = np.argsort(-scores)
```

In […
```
sort_index
```

Out[…
```
array([ 6,  7, 20,  1, 21, 23,  4, 25, 14,  0, 17,  5, 18,  3,  8, 10, 16,
       12,  9, 22, 15, 19, 24, 11, 13,  2], dtype=int64)
```

In […
```
# Resumen desordenado
oraciones = []
cantidad_oraciones = 10
for i in range(cantidad_oraciones):
    oraciones.append([sort_index[i], scores[sort_index[i]], lineas[sort_index[i]]])
    print(f'{scores[sort_index[i]]}:\n{lineas[sort_index[i]]}\n')
```

```
1.0:
 Jones

0.25:
 Él y su esposa han sido frugales durante mucho
tiempo

0.125:
 Está posponiendo la
instalación de un nuevo revestimiento de aluminio en su casa porque
los precios han subido

0.1111111111111111:
se aceleró a una tasa anual de
75 en enero alcanzando un máximo de cuatro décadas

0.09090909090909093:
 Pagó el préstamo de su automóvil hace unos
dos años pero continúa conduciendo un BMW 2014 golpeado

0.07692307692307694:
El fundador de la empresa de ferias comerciales Shamrock
Productions condujo su Oldsmobile durante 450.000 millas hasta que el
motor explotó

0.07692307692307694:
 dos veces por semana
para llenar la camioneta de la familia y tratar de evitar las colas en
la bomba durante la crisis energética de 1979

0.07142857142857144:
 Ella también se está saltando vacaciones costosas
por ahora.A cada paso la gente se ve afectada por el aumento de los
precios dijo la señora Navratil.

0.0666666666666667:
 Cuando cerró la compra de la casa
la tasa hipotecaria que el corredor había ofrecido saltó a cerca de
135 desde alrededor de 1275 que tenía cuando había iniciado el proceso
dijo

0.0666666666666667:
El actual brote de inflación es un momento de déjà vu para las
personas que vivieron las subidas de precios de principios de la
década de 1980.La inflación de EE.UU
```

In […

```python
# Ordenamiento de la lista por el primer elemento de cada sublista
oraciones_sort = sorted(oraciones, key=lambda x:x[0])

#Imprimimos la lista ordenada
for item in oraciones_sort:
    print(item[2]) # el 2 es la columna de las líneas
```

El actual brote de inflación es un momento de déjà vu para las personas que vivieron las subidas de precios de principios de la década de 1980.La inflación de EE.UU se aceleró a una tasa anual de 75 en enero alcanzando un máximo de cuatro décadas dos veces por semana para llenar la camioneta de la familia y tratar de evitar las colas en la bomba durante la crisis energética de 1979 Jones Él y su esposa han sido frugales durante mucho tiempo Cuando cerró la compra de la casa la tasa hipotecaria que el corredor había ofrecido saltó a cerca de 135 desde alrededor de 1275 que tenía cuando había iniciado el proceso dijo Está posponiendo la instalación de un nuevo revestimiento de aluminio en su casa porque los precios han subido Pagó el préstamo de su automóvil hace unos dos años pero continúa conduciendo un BMW 2014 golpeado El fundador de la empresa de ferias comerciales Shamrock Productions condujo su Oldsmobile durante 450.000 millas hasta que el motor explotó Ella también se está saltando vacaciones costosas por ahora.A cada paso la gente se ve afectada por el aumento de los precios dijo la señora Navratil.

Conclusiones

Hace algunos años, la vectorización era la manera más adecuada para generar resúmenes, como podemos notar en los resultados, hace falta un poco de coherencia, pero se puede entender la idea del texto que se pretende resumir.

Mg. Luis Felipe Bustamante Narváez

Anexo 15

Proyecto 15: Resumen de textos con Text Rank

Mg. Luis Felipe Bustamante Narváez

En este ejercicio, desarrollaremos un generador de resumen de textos avanzado, utilizando el método TextRank, el cual permite analizar la similitud de las oraciones y palabras para contextualizar las ideas principales del tecto original.

Recordemos que el TextRank está basado en el PageRank de Google, donde las oraciones o palabras equivalen a las páginas web y la similitud entre las oraciones o palabras, representan los enlaces que estas páginas tienen con otras páginas importantes.

Librerías

In [...
```
pip install networkx
```

In [...
```
import pandas as pd
import numpy as np
import textwrap
import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import textwrap
import networkx as nx
import matplotlib.pyplot as plt
```

In [...
```
nltk.download('punkt')
nltk.download('stopwords')
```

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

Out[... True

Cargamos los Datos

In [...
```
path = 'data/df_total.csv'
df = pd.read_csv(path, encoding='utf-8')
```

In [...
```
df
```

Out[…

	url	news	Type
0	https://www.larepublica.co/redirect/post/3201905	Durante el foro La banca articulador empresari...	Otra
1	https://www.larepublica.co/redirect/post/3210288	El regulador de valores de China dijo el domin...	Regulaciones
2	https://www.larepublica.co/redirect/post/3240676	En una industria históricamente masculina como...	Alianzas
3	https://www.larepublica.co/redirect/post/3342889	Con el dato de marzo el IPC interanual encaden...	Macroeconomia
4	https://www.larepublica.co/redirect/post/3427208	Ayer en Cartagena se dio inicio a la versión n...	Otra
...	...	...	...
1212	https://www.bbva.com/es/como-lograr-que-los-in...	En la vida de toda empresa emergente llega un ...	Innovacion
1213	https://www.bbva.com/es/podcast-como-nos-afect...	La espiral alcista de los precios continúa y g...	Macroeconomia
1214	https://www.larepublica.co/redirect/post/3253735	Las grandes derrotas nacionales son experienci...	Alianzas
1215	https://www.bbva.com/es/bbva-y-barcelona-healt...	BBVA ha alcanzado un acuerdo de colaboración c...	Innovacion
1216	https://www.larepublica.co/redirect/post/3263980	Casi entrando a la parte final de noviembre la...	Alianzas

1217 rows × 3 columns

In […

```
print(df['news'][2][:500])
```

```
En una industria históricamente masculina como lo es la aviación Viva presentó su avión ros
ado A320NEO que apuesta por la equidad de género la lucha contra el cáncer de mama la inclu
sión y la diversidad.Desde Francia llegó Go Pink que tuvo un precio promedio de US50 millon
es convirtiéndose en la aeronave número 20 de las 21 con las que finalizará el año esta aer
olínea.En Viva estamos trabajando muy fuerte para que haya más mujeres. Actualmente el grup
o ejecutivo está compuesto por 42 mujeres per
```

In […

```
# Buscamos una noticia larga para tomar como ejemplo a la hora de hacer el resumen
doc = df['news'].sample()
```

In […

```
print(doc.iloc[0][:500])
```

```
El crecimiento global será en los próximos dos años levemente más bajo a lo esperado pero s
e mantendrá robusto. Tras una caída del 32 en 2020 se estima que el PIB mundial creció alre
dedor de 60 en 2021 y que se expandirá 44 en 2022 y 38 en 2023.El informe de BBVA Research
Situación Argentina 1 trimestre 2022 sostiene que la persistencia de la pandemia y de los p
roblemas en las cadenas de suministro globales además de frenar el crecimiento económico ma
ntendrán la inflación global elevada principa
```

In […

```
# Obtener el índice de la noticia para manipulación de datos
indice = df.index[df['news'] == doc.iloc[0]].tolist()
print(indice)
```

```
[804]
```

In […

```
# hacemos la prueba con un texto que tiene bastante texto
print(df['news'][257][:100])
```

```
Como cambiarle a un coche su motor diésel por uno eléctrico mientras está en marcha. Esta a
nalogía d
```

In […

```
#Buscamos el texto largo que encontramos en una fase de pruebas
doc = df.loc[257, 'news']
```

In […

```
doc[:500]
```

Out[…

```
'Como cambiarle a un coche su motor diésel por uno eléctrico mientras está en marcha. Esta
analogía define la tarea titánica que ha supuesto para Ethereum su última actualización. F
ruto de un trabajo de varios años, ‘The Merge’ constituye un cambio de paradigma sobre cóm
o se generan los criptoactivos de forma más segura y sostenible (elimina hasta un 99,95% d
e la energía necesaria hasta ahora), y permitirá que las finanzas descentralizadas (DeFi)
afronten la implantación a gran escala. Pero tambi'
```

Procesamiento de Datos

TextWrap

In [...
```
# Eliminamos las palabras cortadas de las líneas
doc2 = textwrap.fill(doc, replace_whitespace=False, fix_sentence_endings=True)
```

In [...
```
print(doc2[:500])
```

```
Como cambiarle a un coche su motor diésel por uno eléctrico mientras
está en marcha.  Esta analogía define la tarea titánica que ha
supuesto para Ethereum su última actualización.  Fruto de un trabajo
de varios años, 'The Merge' constituye un cambio de paradigma sobre
cómo se generan los criptoactivos de forma más segura y sostenible
(elimina hasta un 99,95% de la energía necesaria hasta ahora), y
permitirá que las finanzas descentralizadas (DeFi) afronten la
implantación a gran escala.  Pero ta
```

Separación en líneas

In [...
```
# Podemos separar por líneas, por puntos o comas, la idea es conservar oraciones
# con ideas claras.
lineas = doc2.split('. ')  #Usamos punto espacio, por las siglas o números que pueden haber
```

In [...
```
len(lineas)
```

Out[...
```
24
```

In [...
```
lineas[22:]
```

Out[...
```
[' Pero ahora\nGary Gensler, presidente de la SEC (la Comisión de Bolsa y Valores de\nEsta
dos Unidos), ha apuntado a la posibilidad de categorizarlo como un\nvalor, ya que con la f
órmula del PoS se comporta como tal (genera\ndividendos a través del depósito), lo que lo
sometería a las\nnormativas que regulan los valores.\r\nBrett Harrison, presidente de la\n
plataforma de criptoactivos FTX.US, ha recordado que esta complejidad\naumenta al existir
diferentes tipos de participación',
 ' Un usuario\npuede depositar sus 'tokens' para convertirse en validador, participar\nde
manera indirecta a través de un 'exchange', como se ha mencionado\nanteriormente, o presta
rlos en protocolos financieros descentralizados\nen los que obtener también rendimiento
s.\r\nEl debate está sobre la\nmesa.']
```

Eliminación de oraciones vacías

In [...
```
lineas = [item for item in lineas if item.strip()]
```

In [...
```
len(lineas)
```

Out[...
```
24
```

In [...
```
lineas[22:]
```

Out[…
```
[' Pero ahora\nGary Gensler, presidente de la SEC (la Comisión de Bolsa y Valores de\nEsta
dos Unidos), ha apuntado a la posibilidad de categorizarlo como un\nvalor, ya que con la f
órmula del PoS se comporta como tal (genera\ndividendos a través del depósito), lo que lo
sometería a las\nnormativas que regulan los valores.\r\nBrett Harrison, presidente de la\n
plataforma de criptoactivos FTX.US, ha recordado que esta complejidad\naumenta al existir
diferentes tipos de participación',
 ' Un usuario\npuede depositar sus ‘tokens’ para convertirse en validador, participar\nde
manera indirecta a través de un ‘exchange’, como se ha mencionado\nanteriormente, o presta
rlos en protocolos financieros descentralizados\nen los que obtener también rendimiento
s.\r\nEl debate está sobre la\nmesa.']
```

Vectorización

In […
```
# Creamos la tokenización usando las stopwords descargadas
tokenizar = TfidfVectorizer(stop_words=stopwords.words('spanish'), norm='l1')
```

In […
```
# Creamos la matriz
X = tokenizar.fit_transform(lineas)
X
```

Out[…
```
<24x402 sparse matrix of type '<class 'numpy.float64'>'
        with 539 stored elements in Compressed Sparse Row format>
```

In […
```
# Mostramos la matriz
filas, columnas = X.shape

for i in range(10):  #aquí ponemos las filas, pero al ser muchas el resultado es extenso.
    for j in range(10):
        print(X[i, j], end=' ') # imprime el elemento y un espacio en blanco
    print()  #deja el renglón

# Cada fila va a representar una palabra
# Cada columna va a representar cada una de las oraciones
```

```
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.022037911607183422 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0467110969446465 0.0 0.0 0.0 0.0
```

Calculamos la similitud

In […
```
# Matriz de similitud
S = cosine_similarity(X)
```

In […
```
S.shape
```

Out[…
```
(24, 24)
```

In […
```
# Muestra de similitudes, por ejemplo la oración 0, tiene similitud de 0.1248 con la
# oración 6.
S[:1]
```

Out[…
```
array([[1.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.12486586, 0.        , 0.        , 0.12170713,
        0.07641065, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        ]])
```

Normalización

In […
```
S = S / S.sum(axis=1, keepdims=True)
```

In […
```
S[:1]
```

Out[…
```
array([[0.75586724, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.09438202, 0.        , 0.        , 0.09199444,
        0.05775631, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        ]])
```

In […
```
# Probamos
S[0].sum()
```

Out[…
```
1.0
```

Suavizado (Markov)

In […
```
# Matriz de transición uniforme
U = np.ones_like(S)
```

In […
```
U[:1]
```

Out[…
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

In […
```
U.shape
```

Out[…
```
(24, 24)
```

Normalizamos la matriz de transición

In […
```
U = U / len(U)
```

In […
```
U[:1]
```

Out[…
```
array([[0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,
        0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,
        0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,
        0.04166667, 0.04166667, 0.04166667, 0.04166667, 0.04166667,
        0.04166667, 0.04166667, 0.04166667, 0.04166667]])
```

Matriz de similitud suavizada

In […
```
factor = 0.1  #valor pequeño para omitir ceros
S_s = (1 - factor)*S + factor*U
```

In […
```
S_s[:1]
```

Out[…
```
array([[0.68444718, 0.00416667, 0.00416667, 0.00416667, 0.00416667,
        0.00416667, 0.08911048, 0.00416667, 0.00416667, 0.08696166,
        0.05614734, 0.00416667, 0.00416667, 0.00416667, 0.00416667,
        0.00416667, 0.00416667, 0.00416667, 0.00416667, 0.00416667,
        0.00416667, 0.00416667, 0.00416667, 0.00416667]])
```

Explicación

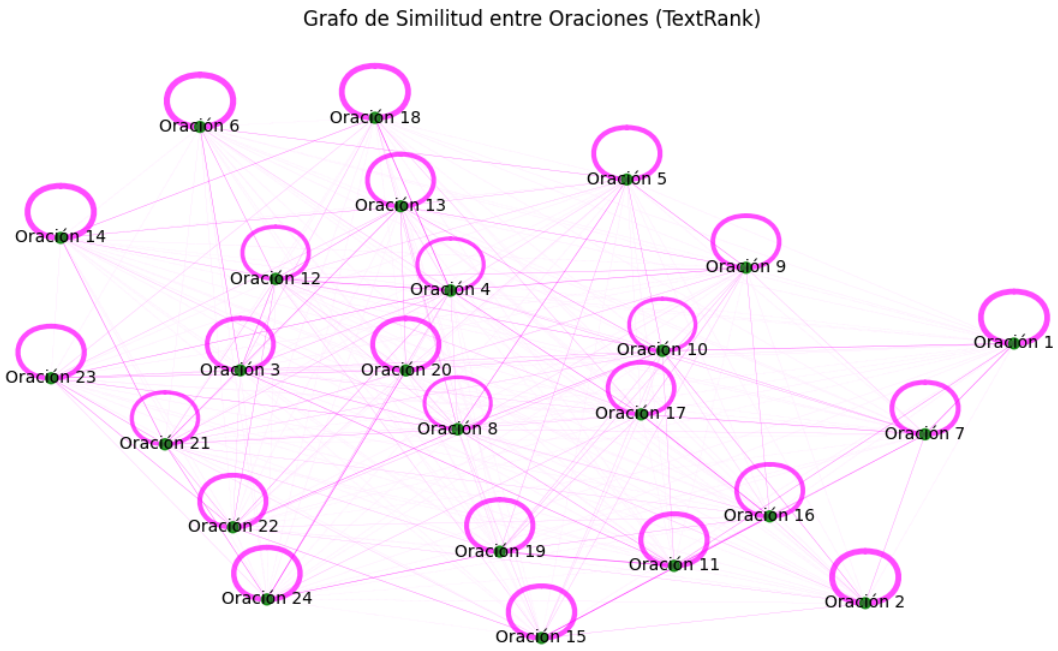
La matriz original contiene valores 0.0 con los cuales es imposible calcular promedios cuando toda una fila tenga estos valores, lo que generará errores de ejecución o desbordamientos en los resultados. En este caso, usaremos el suavizado de Laplace de Markov, agregando un porcentaje mínimo de selección de los datos con la matriz de transición normalizada, luego aplicamos la fórmula S_s = (1 - factor)*S* + *factor*U que utiliza un factor pequeño que dividirá los resultados para este caso en un 90% del total, para usar el 10% restante que modifica los ceros posibles encontrados en cada posición de la matriz.

Grafo

Debemos crearlo por aparte

In […
```python
# Creamos el grafo
G = nx.from_numpy_array(S_s)
scores_G = nx.pagerank(G)
```

In […
```python
# Visualizamos el grafo
plt.figure(figsize=(10,6))
pos = nx.spring_layout(G, seed=42)

# Dibujamos los nodos con tamaño proporcional al score_G
nx.draw_networkx_nodes(G, pos,
                       node_size=[1000 * scores_G[i] for i in G.nodes()],
                       node_color='green',
                       alpha=0.8)

# Dibujar aristas con pesos
edges = G.edges(data=True)
nx.draw_networkx_edges(G, pos,
                       width=[5 * d['weight'] for (_, _, d) in edges],
                       alpha=0.7,
                       edge_color='fuchsia')

# Etiquetas: mostrar la primera parte de cada oración
labels = {i: f'Oración {i+1}' for i in G.nodes()}
nx.draw_networkx_labels(G, pos, labels, font_size=10)

plt.title("Grafo de Similitud entre Oraciones (TextRank)")
plt.axis('off')
plt.tight_layout()
plt.show()
```



Explicación

Se crea un grafo no dirigido usando **G = nx.from_numpy_array(S)**, donde cada nodo representa una oración y cada arista la similitud entre ellas. Posteriormente, se utiliza el **scores = nx.pagerank(G)** para rankear la importancia de cada oración, según como se conecta con otras.

Se crea la figura del pyplot de acuerdo al tamaño del grafo y con el **draw_networkx_nodes** se dimensionan los nodos al tamaño del ranking. Luego, el **draw_networkx_edges**, crea las aristas que interconectan cada nodo.

Finalmente, se crean las etiquetas que aparecerán en cada nodo, para nuestro caso la palabra **'Oración'** y el número del peso de la Matriz de similitud suavizada, para mostrarla a través del **plt.show()**.

TextRank

Matriz Estacionaria

Ya tenemos la matriz de transición, que nos indica la probabilidad de pasar de una oración a otra, ahora, necesitamos saber cuál es la probabilidad del estado actual en el tiempo, es decir de la oración que se está analizando sin tener en en cuenta hacia qué otra oración puede ir.

In [...
```
eigen_values, eigen_vectors = np.linalg.eig(S.T)
```

In [...
```
eigen_values
```

Out[...
```
array([1.        , 0.86664673, 0.84524074, 0.82338475, 0.81472008,
       0.77307318, 0.76045412, 0.74948088, 0.73398798, 0.72032562,
       0.41731513, 0.42223906, 0.45962069, 0.46559459, 0.47809886,
       0.50087105, 0.51123721, 0.65981025, 0.56322085, 0.57564347,
       0.58091814, 0.63872781, 0.61032953, 0.62012229])
```

In [...
```
# Buscamos la posición donde el eigen_values fue 1
pos_eigen = np.where(np.isclose(eigen_values, 1.0))[0]
print(pos_eigen[0])
```

0

In [...
```
# Localizamos el eigen_vector de la posición donde halló el 1.0
eigen_vectors[:,pos_eigen[0]]
```

Out[...
```
array([-0.17088315, -0.16683293, -0.20567838, -0.2498079 , -0.19706824,
       -0.15889722, -0.19988734, -0.26248927, -0.20897602, -0.24820384,
       -0.19229987, -0.2317503 , -0.2060358 , -0.16720484, -0.18848506,
       -0.21172529, -0.21567576, -0.16973832, -0.19770387, -0.19485202,
       -0.23640176, -0.20003379, -0.19189516, -0.18299742])
```

Explicación

📌 1. ¿Qué es np.linalg.eig()? Es una función de NumPy (np) que calcula:

- Los valores propios (eigenvalues): escalares λ
- Los vectores propios (eigenvectors): vectores v

...de una matriz cuadrada A, tales que:

$$Av=\lambda v$$

📌 2. ¿Qué hace S.T? S.T es simplemente la transpuesta de la matriz S. Si S es una matriz de tamaño $m \times n$, entonces ST es de tamaño $n \times m$.

Este paso puede ser necesario si, por ejemplo, queremos que la matriz sea cuadrada para aplicar la descomposición (ya que solo matrices cuadradas tienen valores/vectores propios bien definidos).

📐 ¿Qué significan los eigenvalores y eigenvectores?

- Eigenvalor λ: Indica cuánto se estira o encoge un vector propio al aplicarle la transformación A.
- Eigenvector v: Un vector que no cambia de dirección bajo la transformación A, solo cambia su magnitud.

Puntuación

In [...
```
scores = eigen_vectors[:,pos_eigen[0]]
```

In [...
```
sort_index = np.argsort(-scores)
```

In [...
```
# orden de oraciones más importantes
sort_index
```

Out[...
```
array([ 5,  1, 13, 17,  0, 23, 14, 22, 10, 19,  4, 18,  6, 21,  2, 12,  8,
       15, 16, 11, 20,  9,  3,  7], dtype=int64)
```

Resumen

In [...
```
print('Resumen\n')
cantidad_oraciones = 6
for i in sort_index[:cantidad_oraciones]:
    print("\n".join(textwrap.wrap(f"{scores[i]:.2f}: {lineas[i]}", width=50)))
```

Resumen

-0.16: Un método muy seguro y probado aunque poco eficiente desde el punto de vista sostenible, ya que exige usar equipos computacionales cada vez más potentes que son grandes consumidores de energía
-0.17: Esta analogía define la tarea titánica que ha supuesto para Ethereum su última actualización
-0.17: En caso de conductas maliciosas, se podría penalizar al atacante con parte o la totalidad de los ethers que bloqueó en un inicio
-0.17: Como el hilo de Ariadna, este ‘criptoglosario’ muestra cómo se relacionan unos con otros en la economía que viene… o que ya está aquí. Como contrapartida, preocupa la pérdida de descentralización que podría acarrear PoS
-0.17: Como cambiarle a un coche su motor diésel por uno eléctrico mientras está en marcha
-0.18: Un usuario puede depositar sus ‘tokens’ para convertirse en validador, participar de manera indirecta a través de un ‘exchange’, como se ha mencionado anteriormente, o prestarlos en protocolos financieros descentralizados en los que obtener también rendimientos. El debate está sobre la mesa.

Conclusiones

La tecnología del método TextRank, evidencia un avance significativo en la realización de resúmenes. Podemos observar la coherencia del texto y como se comprende fácilmente la idea principal de la noticia que se tomó como ejemplo.

Mg. Luis Felipe Bustamante Narváez

Anexo 16

Proyecto 16: Modelado de Temas con LDA

Mg. Luis Felipe Bustamante Narváez

En este ejercicio, desarrollaremos un clustering de temas de acuerdo con un dataset que contiene transcripciones de entrevistas, bastante extensas y en español. Se quiere desarrollar un clasificador que permita identificar los temas que se trabajan en cada transcripción de manera que se pueda predecir de qué habla cada una de ellas, y qué afinidad tiene con otras.

Recordemos que, **Clustering** (o agrupamiento) es una técnica de aprendizaje no supervisado cuyo objetivo es agrupar objetos similares entre sí y diferentes de los de otros grupos, sin tener etiquetas previas.

Librerías

In [...]
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
import textwrap
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from tqdm import tqdm
```

Manejo de StopWords

In [...]
```
nltk.download('stopwords')
```

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

Out[...] True

In [...]
```
stop_words_original = set(stopwords.words('spanish'))
```

In [...]
```
# Convertimos el conjunto en lista
_stop_words_original = list(stop_words_original)
_stop_words_original[:10]
```

Out[…
```
['sobre',
 'habíais',
 'estarían',
 'unos',
 'por',
 'estás',
 'tienen',
 'estabais',
 'eran',
 'habríamos']
```

Explicación

En una entrevista, de acuerdo con los datos que vamos a analizar, se presentan palabras adicionales a las que un texto pueda llegar a tener, palabras relacionadas con preguntas, exclamaciones, entre otras, que se repiten constantemente y que no aportan al procesamiento de datos.

Por este motivo, adicionaremos algunas palabras observadas en las entrevistas para mejorar el proceso.

In […
```
path_json = 'data/stop_words_news.json'
stop_words_news = pd.read_json(path_json, encoding='utf-8')
```

In […
```
stop_words_news[:10]
```

Out[…

	words
0	así
1	si
2	hacer
3	cosas
4	creo
5	cómo
6	solo
7	aquí
8	risas
9	ser

In […
```
stop_words_news = set(stop_words_news['words'])
```

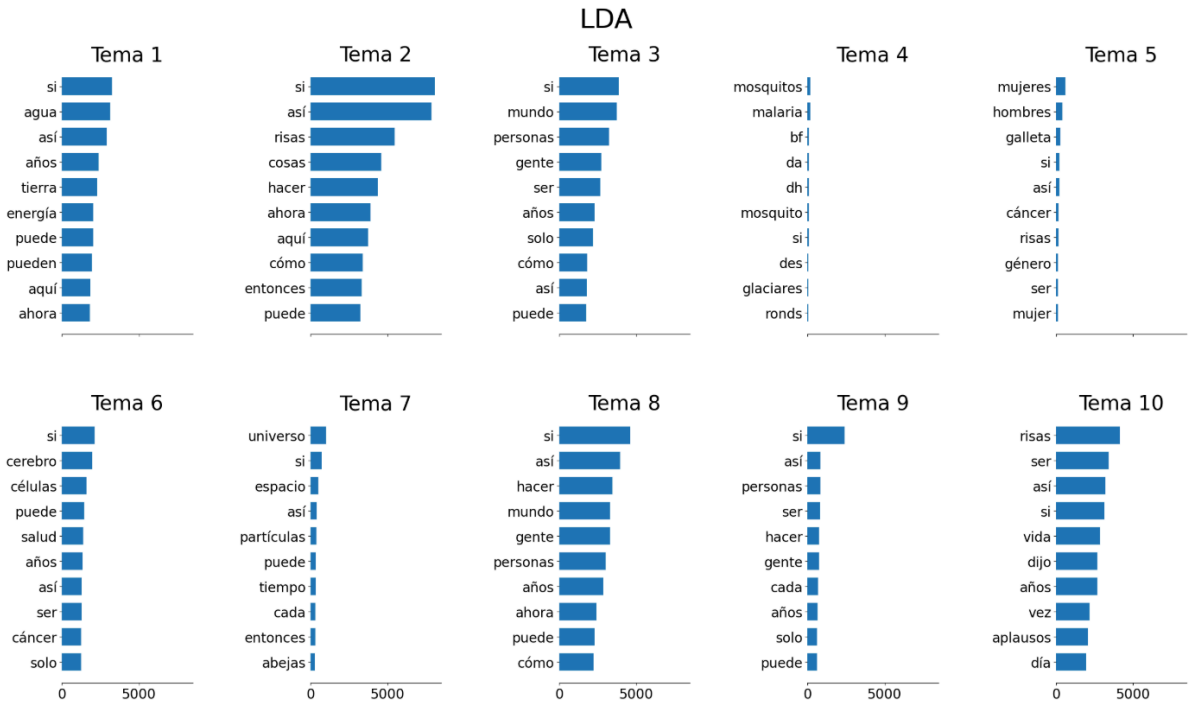
In […
```
stop_words_add = stop_words_original.union(stop_words_news)
```

In […
```
_stop_words_add = list(stop_words_add)
_stop_words_add[:10]
```

Out[…
```
['sobre',
 'habíais',
 'estarían',
 'ser',
 'sentido',
 'hombre',
 'unos',
 'fuésemos',
 'habidas',
 'por']
```

Nota: *Primero realizaremos el proceso con las stopwords de la librería y luego con las que hemos añadido, de tal manera poder observar si hay diferencias en el clustering.*

Resultado del LDA con las palabras originales de la librería



Posteriormente, se ejecutó con palabras añadidas; este es el producto real esperado del LDA, mostrado en las siguientes celdas de código.

Cargamos los Datos

In [...
```
path = 'data/ted_talks_es.csv'
df = pd.read_csv(path)
```

In [...
```
df.iloc[:4,16:]
```

Out[...

	url	description	transcript
0	https://www.ted.com/talks/al_gore_averting_the...	Con el mismo humor y humanidad que irradió en ...	Muchas gracias Chris. Y es en verdad un gran h...
1	https://www.ted.com/talks/david_pogue_simplici...	El columnista del New York Times, David Pogue,...	Hola contestadora automática, mi vieja amiga. ...
2	https://www.ted.com/talks/majora_carter_greeni...	En una charla altamente emotiva, la activista ...	Si están presentes aquí hoy, y estoy muy conte...
3	https://www.ted.com/talks/sir_ken_robinson_do_...	Sir Ken Robinson plantea de manera entretenida...	Buenos días. ¿Cómo están? Ha sido increíble, ¿...

In [...
```
df['transcript'][0][:500]
```

Out[...
```
'Muchas gracias Chris. Y es en verdad un gran honor tener la oportunidad de venir a es
te escenario por segunda vez. Estoy extremadamente agradecido. He quedado conmovido po
r esta conferencia, y deseo agradecer a todos ustedes sus amables comentarios acerca d
e lo que tenía que decir la otra noche. Y digo eso sinceramente, en parte porque — (So
llozos fingidos) — ¡lo necesito! (Risas) ¡Pónganse en mi posición! Volé en el avión vi
cepresidencial por ocho años. ¡Ahora tengo que quitarme mis zapatos o b'
```

Procesamiento de los Datos

Vectorización

In [...
```
# Cada vector es una conferencia y cada conferencia tiene vectores de cada palabra
vectorizer = CountVectorizer(stop_words=_stop_words_add)
```

In [...
```
# Aplicamos la vectorización a nuestro dataset
X = vectorizer.fit_transform(df['transcript'])
X
```

Out[...
```
<3921x111271 sparse matrix of type '<class 'numpy.int64'>'
        with 1889073 stored elements in Compressed Sparse Row format>
```

Modelo LDA

In [...
```
lda = LatentDirichletAllocation(
    n_components=10,  #default:10
    random_state=12354,  #estados aleatorios - semilla de la forma en que comienzan
)
```

In [...
```
lda.fit(X)
```

Out[...
```
LatentDirichletAllocation(random_state=12354)
```

Gráfico de Palabras TOP

In [...
```
def graficar_palabras_top(model, feature_names, n_top_words=10):
    fig, axes = plt.subplots(2, 5, figsize=(30, 15), sharex=True)
    axes = axes.flatten()
    for topic_index, topic in enumerate(model.components_):
        top_features_ind = topic.argsort()[: -n_top_words - 1 : -1]
        top_features = [feature_names[i] for i in top_features_ind]
        weights = topic[top_features_ind]

        ax = axes[topic_index]
        ax.barh(top_features, weights, height=0.7)
        ax.set_title(f'Tema {topic_index + 1}', fontdict={'fontsize': 30})
        ax.invert_yaxis()
        ax.tick_params(axis='both', which='major', labelsize= 20)
        for i in 'top right left'.split():
            ax.spines[i].set_visible(False)
        fig.suptitle('LDA', fontsize= 40)
    plt.subplots_adjust(top= 0.90, bottom=0.05, wspace=0.90, hspace=0.3)
    plt.show()
```

Explicación

La función **graficar_palabras_top** sirve para visualizar las palabras más importantes de cada tema generado por un modelo **LDA**, que es comúnmente usado en procesamiento de lenguaje natural para descubrir temas latentes en un conjunto de textos.

Primero, en la definición de la función, se reciben tres parámetros: el modelo **LDA** ya entrenado, una lista de nombres de las palabras (que suelen provenir de un **CountVectorizer** o **TfidfVectorizer**), y un número opcional que indica cuántas palabras principales por tema se quieren graficar (por defecto **10**).

Luego, se crea una figura con **10 subgráficas** organizadas en 2 filas y 5 columnas, lo cual permite visualizar hasta 10 temas. Estas subgráficas se aplanan usando **flatten()** para que sea más fácil iterar sobre ellas.

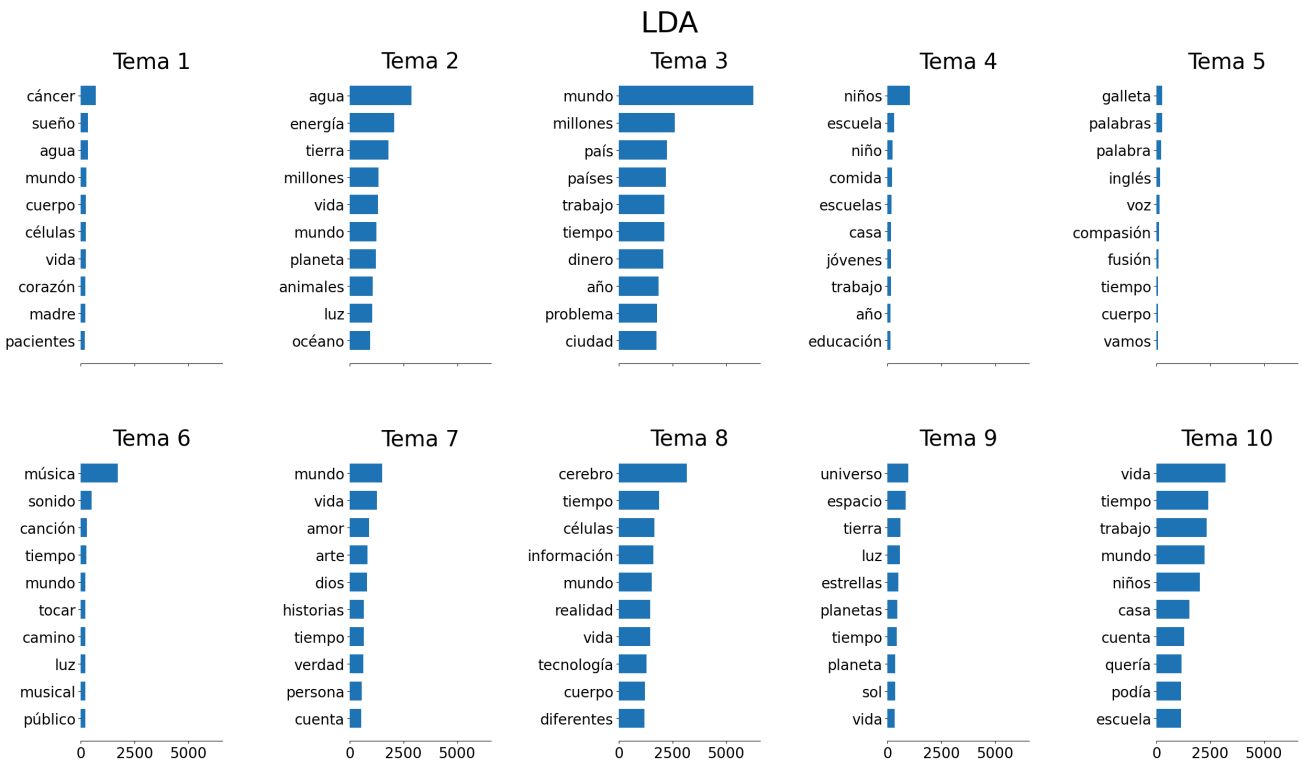
Después, comienza un ciclo con **enumerate** que recorre cada tema del modelo. Cada tema se representa como un vector que contiene los pesos (importancia) de cada palabra. Se ordenan estos pesos con **argsort()** para seleccionar las palabras más relevantes del tema, tomando sus índices. Luego, se usan esos índices para obtener tanto los nombres de las palabras como sus respectivos pesos.

Con esa información, se crea un gráfico de barras horizontal usando **barh()** en la subgráfica correspondiente, donde las palabras están en el eje Y y sus pesos en el eje X. Se le da un título a cada gráfico con el número del tema usando **set_title**, se invierte el eje Y con **invert_yaxis()** para que la palabra más importante esté arriba, y se ajustan los tamaños de fuente con **tick_params** para que sea más legible. También se eliminan algunos bordes del gráfico con **set_visible(False)** para que se vea más limpio.

Finalmente, se añade un título general a toda la figura con **suptitle()**, se ajusta el espacio entre los gráficos con **subplots_adjust()**, y se muestra el resultado con **plt.show()**.

En resumen, esta función te ayuda a interpretar de manera visual qué palabras definen cada tema extraído por el modelo LDA, lo cual es clave para entender los patrones latentes en un conjunto de textos.

In [...
```
# Obtenemos las palabras
palabras = vectorizer.get_feature_names_out()
# llamamos la función para graficar
graficar_palabras_top(lda, palabras);
```



Prueba del Modelo LDA

Seleccionamos un tema al azar de toda la base de datos, y el modelo debe indicarnos de qué tema está hablando.

Matriz transformada

In [...

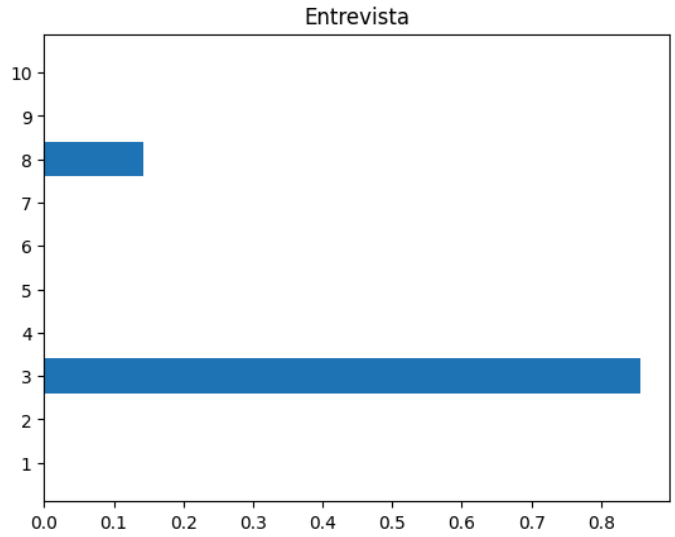
```
Z = lda.transform(X)
```

Solución gráfica

In [...

```
np.random.seed(1111)  # semilla fija
i = np.random.choice(len(df))
z = Z[i]
topics = np.arange(10) + 1

fig, ax = plt.subplots()
ax.barh(topics, z)
ax.set_yticks(topics)
ax.set_title('Entrevista');
print(f'Entrevista seleccionada al azar, número {i}')
```

```
Entrevista seleccionada al azar, número 2460
```



Tema de la entrevista seleccionada - Comparación

In [...
```
def wrap(x):
    entrevista = textwrap.fill(x,
                               replace_whitespace=False,
                               fix_sentence_endings=True)
    return entrevista
```

In [...
```
print(wrap(df.iloc[i]['transcript'][:500]))
```

```
Hoy voy a hablar de tecnología y de la sociedad.  El Departamento de
Transporte estimó que, el año pasado, hubo 35 000 muertos en
accidentes de auto, tan solo en EE.UU. A nivel mundial, mueren 1,2
millones de personas por año en accidentes de auto.  Si hubiera una
manera de eliminar el 90 % de esos accidentes, ¿apoyarían la causa?
Por supuesto que lo harían.  Esto es lo que promete la tecnología de
vehículos autónomos, al eliminar la principal causa de accidentes: el
error humano.  Imagínense en el
```

Conclusiones

LDA, permite crear un clustering de temas y palabras asociadas a la información que trae el texto por defecto, generando un proceso estadístico capaz de filtrar la información para ofrecer mejores interpretaciones en el procesamiento de documentos.

Mg. Luis Felipe Bustamante Narváez

In [...