2023

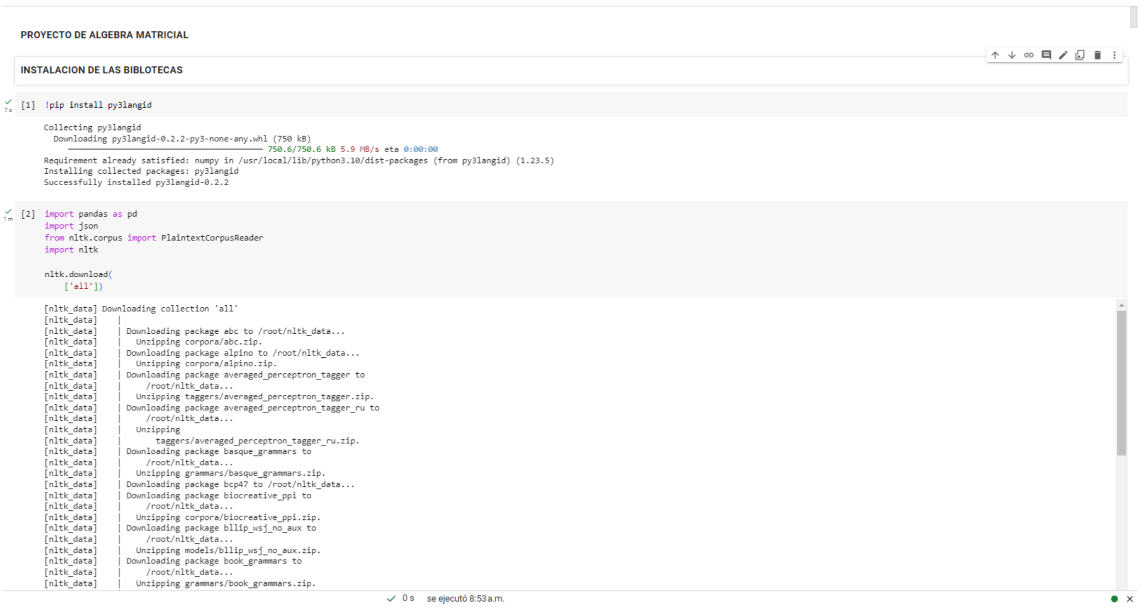
CLASIFICADOR DE COMENTARIOS SPAM Y NO SPAM

GRUPO 1 – ALGEBRA MATRICIAL

KATERINE LISBETH RAFAEL BOURDIERD 2022-0088
WINIFEL FELICIA CALCAÑO FULGENCIO 2019-7734
HAZER SCOTT – 20221684
ISAAC RAMIREZ - 2021-2313
EDUARDO SILFA - 2021-1949

PROYECTO FINAL DE ALGEBRA MATRICIAL

Link: https://colab.research.google.com/drive/1Nlq3b7kB-pM91WxiFa9VgkP8LgB-s4w6?usp=sharing



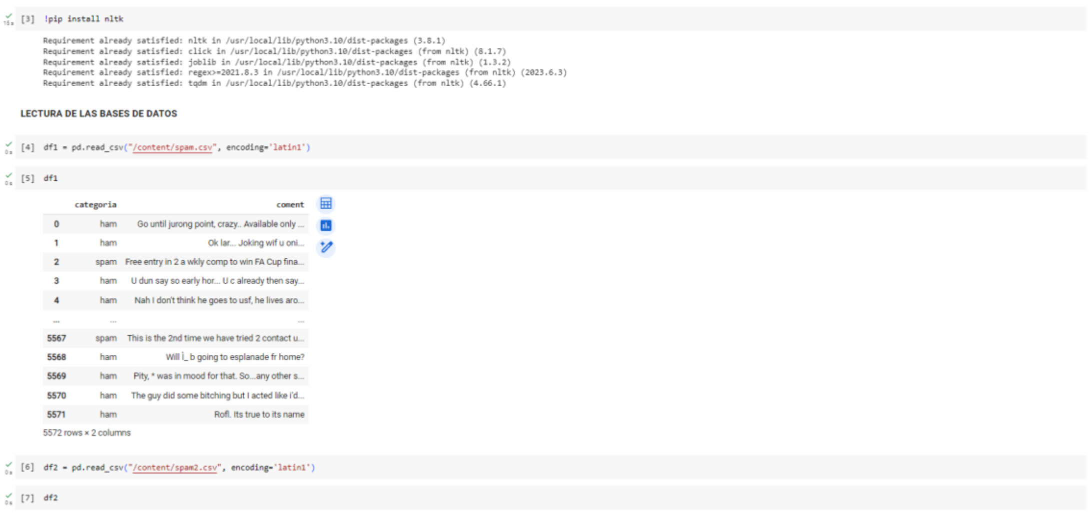
PROYECTO DE ALGEBRA MATRICIAL

INSTALACION DE LAS BIBLOTECAS

```
[1] !pip install py3langid

    Collecting py3langid
      Downloading py3langid-0.2.2-py3-none-any.whl (750 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 750.6/750.6 kB 5.9 MB/s eta 0:00:00
    Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from py3langid) (1.23.5)
    Installing collected packages: py3langid
    Successfully installed py3langid-0.2.2
```

```
[2] import pandas as pd
    import json
    from nltk.corpus import PlaintextCorpusReader
    import nltk

    nltk.download(
        ['all'])

    [nltk_data] Downloading collection 'all'
    [nltk_data]    |
    [nltk_data]    | Downloading package abc to /root/nltk_data...
    [nltk_data]    |   Unzipping corpora/abc.zip.
    [nltk_data]    | Downloading package alpino to /root/nltk_data...
    [nltk_data]    |   Unzipping corpora/alpino.zip.
    [nltk_data]    | Downloading package averaged_perceptron_tagger to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
    [nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping
    [nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
    [nltk_data]    | Downloading package basque_grammars to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping grammars/basque_grammars.zip.
    [nltk_data]    | Downloading package bcp47 to /root/nltk_data...
    [nltk_data]    | Downloading package biocreative_ppi to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
    [nltk_data]    | Downloading package bllip_wsj_no_aux to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
    [nltk_data]    | Downloading package book_grammars to
    [nltk_data]    |     /root/nltk_data...
    [nltk_data]    |   Unzipping grammars/book_grammars.zip.
```

✓ 0 s se ejecutó 8:53 a.m.



```
[3] !pip install nltk

    Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
    Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
    Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
    Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
    Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

LECTURA DE LAS BASES DE DATOS

```
[4] df1 = pd.read_csv("/content/spam.csv", encoding='latin1')
```

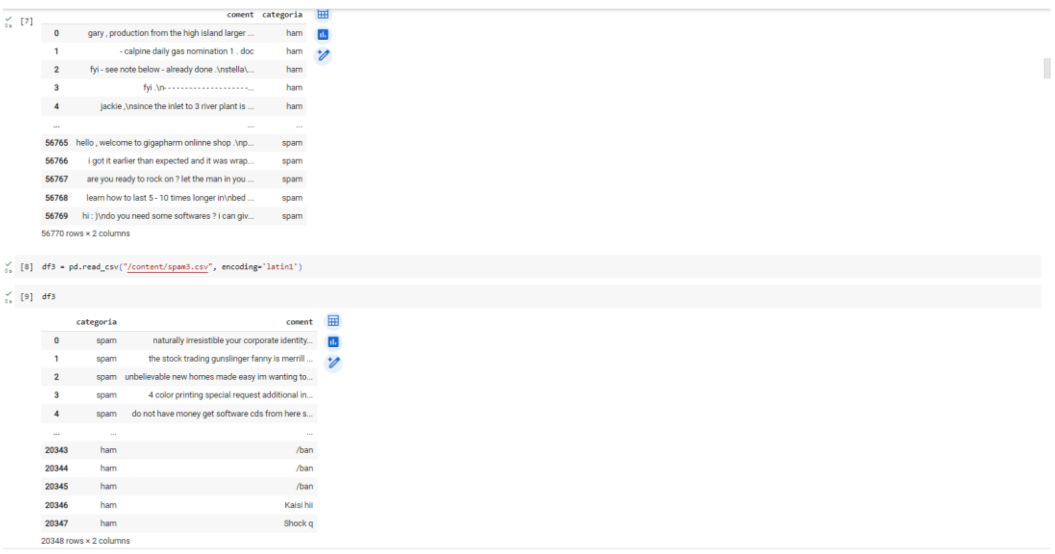
```
[5] df1
```

	categoria	coment
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

```
[6] df2 = pd.read_csv("/content/spam2.csv", encoding='latin1')
```

```
[7] df2
```



[7]

	coment	categoria
0	gary , production from the high island larger ...	ham
1	- calpine daily gas nomination 1 . doc	ham
2	fyi - see note below - already done .\nstella\...	ham
3	fyi .\n- - - - - - - - - - - - - - - - - - - -...	ham
4	jackie ,\nsince the inlet to 3 river plant is ...	ham
...	...	...
56765	hello , welcome to gigapharm onlinne shop .\np...	spam
56766	i got it earlier than expected and it was wrap...	spam
56767	are you ready to rock on ? let the man in you ...	spam
56768	learn how to last 5 - 10 times longer in\nbed ...	spam
56769	hi :)\ndo you need some softwares ? i can giv...	spam

56770 rows × 2 columns

```
[8] df3 = pd.read_csv("/content/spam3.csv", encoding='latin1')
```

```
[9] df3
```

	categoria	coment
0	spam	naturally irresistible your corporate identity...
1	spam	the stock trading gunslinger fanny is merrill ...
2	spam	unbelievable new homes made easy im wanting to...
3	spam	4 color printing special request additional in...
4	spam	do not have money get software cds from here s...
...	...	...
20343	ham	/ban
20344	ham	/ban
20345	ham	/ban
20346	ham	Kaisi hii
20347	ham	Shock q

20348 rows × 2 columns

UNION DE LAS BASES DE DATOS

```
[10] result_df = pd.concat([df1, df2, df3], ignore_index=True)
```

```
[11] result_df
```

	categoria	coment
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
82685	ham	/ban
82686	ham	/ban
82687	ham	/ban
82688	ham	Kaisi hii
82689	ham	Shock q

82690 rows × 2 columns

BUSQUEDA Y ELEIMINACION DE DATOS NULL

```
[12] result_df.isnull().sum()
```

```
categoria    19945
coment       17255
dtype: int64
```

```
[13] df_sin_nulos = result_df.dropna()
```

```
[14] df_sin_nulos
```

[14]

	categoria	coment
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
82685	ham	/ban
82686	ham	/ban
82687	ham	/ban
82688	ham	Kaisi hii
82689	ham	Shock q

62375 rows × 2 columns

ELIMINACION ACERTADA

```
[15] df_sin_nulos.isnull().sum()
```

```
categoria    0
coment       0
dtype: int64
```

```
[16] df_sin_nulos["categoria"].value_counts()
```

```
ham                                                                                            35578
spam                                                                                           23610
 "" he said .                                                                                     15
2001                                                                                               9
 he said .                                                                                         9
                                                                                                 ...
 everything worked like clockwork .                                                                1
 say                                                                                               1
 but stand - alone funds with the sharp focus that a corporate venturing outfit has are fairly few and far between .    1
 we could be very helpful in the interest of the u . s . ""                                        1
 who traded                                                                                        1
Name: categoria, Length: 2494, dtype: int64
```

```
[17] categorias_filtradas = df_sin_nulos[df_sin_nulos["categoria"].isin(["ham", "spam"])]
     conteo_categorias = categorias_filtradas["categoria"].value_counts()

     print(conteo_categorias)
```

```
ham     35578
spam    23610
Name: categoria, dtype: int64
```

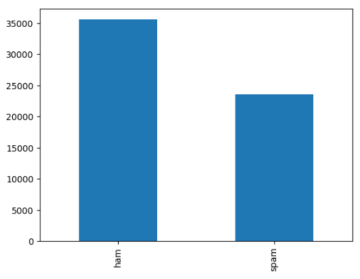
```
[18] df_filtrado = df_sin_nulos[df_sin_nulos["categoria"].isin(["ham", "spam"])]

     df_filtrado["categoria"].value_counts()
```

```
ham     35578
spam    23610
Name: categoria, dtype: int64
```

COMPARACION DE LOS DATOS PARA VER SI EXISTE EL SESGO

```
[19] df_filtrado['categoria'].value_counts().plot.bar()
```

```
<Axes: >
```



```
[20] # Sample 23610 items from the "ham" category
     no_spam = df_filtrado[df_filtrado["categoria"] == "ham"].sample(23610)
     # Sample 23610 items from the "spam" category with replacement
     spam = df_filtrado[df_filtrado["categoria"] == "spam"].sample(23610, replace=True)
```

```
[21] df_filtrado = pd.concat([no_spam, spam])
     df_filtrado = df_filtrado.sample(frac=1).reset_index(drop=True) #Dessordenar las filas
```

LIMPIEZA COMPLETADA Y BASE DE DATOS LIMPIA

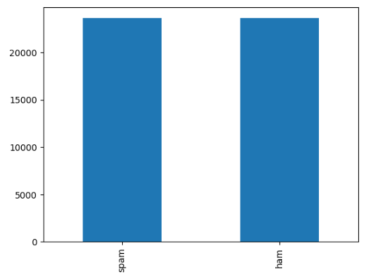
```
[22] #Termina Winnifer
     df_filtrado
```

	categoria	coment
0	spam	hyperlink hyperlink hyperlink let mortgage len...
1	ham	i wan to download nfs most wanted konse site s...
2	ham	fwd enron stanford program content transfer en...
3	spam	all the lastest from stereophonics marley dizz...
4	spam	how to save on aslant your medlcatlons over 60...
...	...	...
47215	spam	guaranteed 50 000 fast you make a guaranteed 5...
47216	ham	daren , i think i may have lost my mind (no c...
47217	spam	ros group presents new residential project 2/3...
47218	ham	shipper\ndaren wanted me to make this request ...
47219	spam	digital convergence . it ' s here , now !\ntoo...

47220 rows × 2 columns

```
[23] df_filtrado['categoria'].value_counts().plot.bar()
```

```
<Axes: >
```



```
[24] #Eduardo
     #Estudio del lenguaje

     import py3langid as langid

     df_filtrado["Language"] = df_filtrado['coment'].apply(lambda x : langid.classify(x)[0])
```

```
[25] dff = df_filtrado [df_filtrado["Language"] == "en"][["coment", "categoria"]]
```

BASE DE DATOS PRINCIPAL

```
[26] dff
```

	coment	categoria
0	hyperlink hyperlink hyperlink let mortgage len...	spam
1	i wan to download nfs most wanted konse site s...	ham
2	fwd enron stanford program content transfer en...	ham
3	all the lastest from stereophonics marley dizz...	spam
4	how to save on aslant your medlcatlons over 60...	spam
...	...	...
47215	guaranteed 50 000 fast you make a guaranteed 5...	spam
47216	daren , i think i may have lost my mind (no c...	ham
47217	ros group presents new residential project 2/3...	spam
47218	shipper\ndaren wanted me to make this request ...	ham
47219	digital convergence . it ' s here , now !\ntoo...	spam

44117 rows × 2 columns

PROCESAMIENTO DEL TEXTO

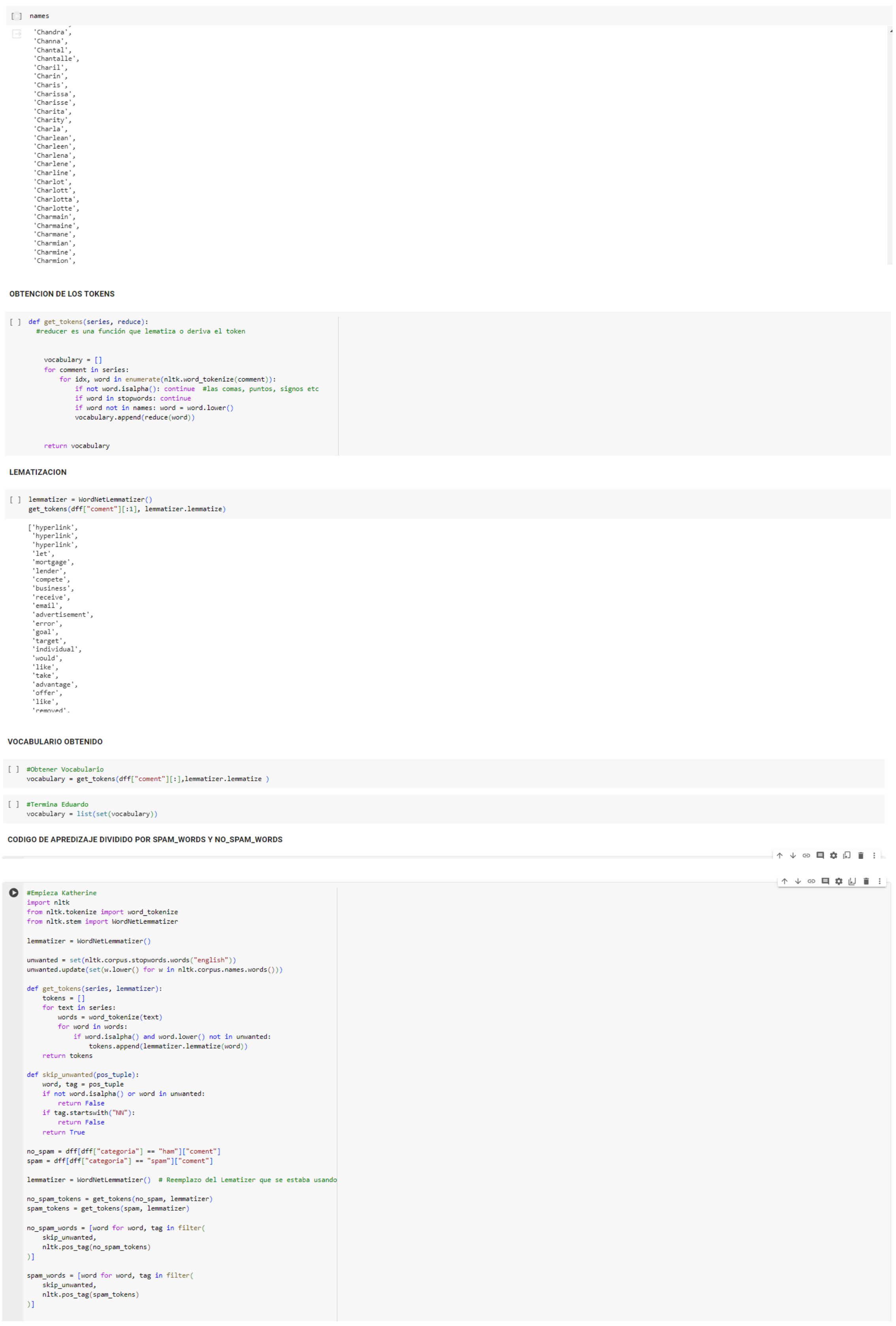
```
#Descargamos las librerias
import nltk
from nltk import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
nltk.download([
    "stopwords", #las stopwords
     "names",    #los nombres
     "vader_lexicon",
     "punkt",
     "wordnet" ])
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package names to /root/nltk_data...
[nltk_data]   Package names is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
[ ] #Obtener las Stopwords del ingles y los names
    stopwords = nltk.corpus.stopwords.words("english")
    names = nltk.corpus.names.words()
```

```
[ ] stopwords
```

```
 'nor',
 'not',
 'only',
 'own',
 'same',
 'so',
 'than',
 'too',
 'very',
 's',
 't',
 'can',
 'will',
```



```
[ ] names
```

```
 'Chandra',
 'Channa',
 'Chantal',
 'Chantalle',
 'Charil',
 'Charin',
 'Charis',
 'Charissa',
 'Charisse',
 'Charita',
 'Charity',
 'Charla',
 'Charlean',
 'Charleen',
 'Charlena',
 'Charlene',
 'Charline',
 'Charlot',
 'Charlott',
 'Charlotta',
 'Charlotte',
 'Charmain',
 'Charmaine',
 'Charmane',
 'Charmian',
 'Charmine',
 'Charmion',
```

OBTENCION DE LOS TOKENS

```
[ ] def get_tokens(series, reduce):
      #reducer es una función que lematiza o deriva el token


        vocabulary = []
        for comment in series:
            for idx, word in enumerate(nltk.word_tokenize(comment)):
                if not word.isalpha(): continue  #las comas, puntos, signos etc
                if word in stopwords: continue
                if word not in names: word = word.lower()
                vocabulary.append(reduce(word))


        return vocabulary
```

LEMATIZACION

```
[ ] lemmatizer = WordNetLemmatizer()
    get_tokens(dff["coment"][:1], lemmatizer.lemmatize)
```

```
['hyperlink',
 'hyperlink',
 'hyperlink',
 'let',
 'mortgage',
 'lender',
 'compete',
 'business',
 'receive',
 'email',
 'advertisement',
 'error',
 'goal',
 'target',
 'individual',
 'would',
 'like',
 'take',
 'advantage',
 'offer',
 'like',
 'removed',
```

VOCABULARIO OBTENIDO

```
[ ] #Obtener Vocabulario
    vocabulary = get_tokens(dff["coment"][:],lemmatizer.lemmatize )
```

```
[ ] #Termina Eduardo
    vocabulary = list(set(vocabulary))
```

CODIGO DE APREDIZAJE DIVIDIDO POR SPAM_WORDS Y NO_SPAM_WORDS

```
#Empieza Katherine
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

unwanted = set(nltk.corpus.stopwords.words("english"))
unwanted.update(set(w.lower() for w in nltk.corpus.names.words()))

def get_tokens(series, lemmatizer):
    tokens = []
    for text in series:
        words = word_tokenize(text)
        for word in words:
            if word.isalpha() and word.lower() not in unwanted:
                tokens.append(lemmatizer.lemmatize(word))
    return tokens

def skip_unwanted(pos_tuple):
    word, tag = pos_tuple
    if not word.isalpha() or word in unwanted:
        return False
    if tag.startswith("NN"):
        return False
    return True

no_spam = dff[dff["categoria"] == "ham"]["coment"]
spam = dff[dff["categoria"] == "spam"]["coment"]

lemmatizer = WordNetLemmatizer()  # Reemplazo del Lematizer que se estaba usando

no_spam_tokens = get_tokens(no_spam, lemmatizer)
spam_tokens = get_tokens(spam, lemmatizer)

no_spam_words = [word for word, tag in filter(
    skip_unwanted,
    nltk.pos_tag(no_spam_tokens)
)]

spam_words = [word for word, tag in filter(
    skip_unwanted,
    nltk.pos_tag(spam_tokens)
)]
```

```
[ ] no_spam_words
```

```
 'continue',
 'sometime',
 'intranet',
 'dear',
 'thanks',
 'already',
 'contacted',
 'update',
 'much',
 'except',
 'publish',
 'maybe',
 'add',
 'heading',
 'really',
 'would',
 'great',
 'get',
 'content',
 'well',
 'explained',
 'various',
 'like',
 'high',
 'probably',
 'written',
 'specifically',
 'purpose',
 'put',
 'interesting',
 'together',
 'pas',
 'collate',
 'update',
 'ect',
 'ec',
```

```
spam_words
```

```
 'making',
 'easier',
 'implies',
 'hydrocodone',
 'attached',
 'u',
 'disturbing',
 'middle',
 'american',
 'hooked',
 'opium',
 'derivative',
 'anxious',
 'non',
 'addictive',
 'gave',
 'extensive',
 'new',
 'like',
 'vicodin',
 'opiate',
 'effective',
 'antitussive',
 'cough',
 'effective',
 'analgesic',
 'moderate',
 'five',
 'equivalent',
 'administered',
 'orally',
 'mg',
 'considered',
 'equivalent',
 'considered',
 'like',
 'semisynthetic',
 'narcotic',
```

OBTENCION DE LAS 200 PALABRAS MAS COMUNES EN SPAM Y NO_SPAM

```
[ ] from pandas.core import common
    spam_fd = nltk.FreqDist(spam_words)
    no_spam_fd = nltk.FreqDist(no_spam_words)

    common_set = set(spam_fd).intersection(no_spam_fd)

    for word in common_set:
        del spam_fd[word]
        del no_spam_fd[word]

    top_200_spam = {word for word, count in spam_fd.most_common(200)}
    top_200_no_spam = {word for word, count in no_spam_fd.most_common(200)}
```

```
[ ] import pickle
    #Pickle s el proceso de convertir un objeto de Python en un flujo de bytes
    #para almacenarlo en un archivo/base de datos
    f = open('top_200_spam.pickle', 'wb')
    pickle.dump(top_200_spam, f) #Pickle se utliza para almacenar
    f.close()

    f = open('top_200_no_spam.pickle', 'wb')
    pickle.dump(top_200_no_spam, f)
    f.close()
```

```
[ ] top_200_spam
```

```
 'powerquest',
 'prescription',
 'presentiy',
 'professiona',
 'professionai',
 'projection',
 'prozac',
 'pubiic',
 'pubiisher',
 'quickbooks',
 'reiated',
 'reieases',
 'reiiable',
 'relaxant',
 'remitted',
```

```
top_200_no_spam
```

```
 'rc',
 'reportedly',
 'restructuring',
 'rpm',
 'rto',
 'saou',
 'savita',
 'sb',
 'sempra',
 'sevil',
 'seyfried',
 'shalesh',
 'shankman',
 'sharad',
 'sherriff',
 'shively',
 'sitara',
 'sj',
 'skilling',
 'socal',
 'sogomonian',
 'soussan',
 'spradling',
```

NLTK Pretrained Sentiment Analyzer

```
[ ] from nltk.sentiment.vader import SentimentIntensityAnalyzer
    #Es el proceso de determinar 'computacionalmente' si un comentario es spam o no_spam
    sia = SentimentIntensityAnalyzer() #Inizializando Sentiment Intensity Analyzer
```

```
[ ] dff["categoria"].value_counts()
```

```
ham     22487
spam    21630
Name: categoria, dtype: int64
```

```
[ ] dff[["coment", "categoria"]]
```

	coment	categoria
0	hyperlink hyperlink hyperlink let mortgage len...	spam
1	i wan to download nfs most wanted konse site s...	ham
2	fwd enron stanford program content transfer en...	ham
3	all the lastest from stereophonics marley dizz...	spam
4	how to save on aslant your medlcatlons over 60...	spam
...	...	...
47215	guaranteed 50 000 fast you make a guaranteed 5...	spam
47216	daren , i think i may have lost my mind (no c...	ham
47217	ros group presents new residential project 2/3...	spam
47218	shipper\ndaren wanted me to make this request ...	ham
47219	digital convergence . it ' s here , now !\ntoo...	spam

44117 rows × 2 columns

NLTK Naive Bayes Classifier

```
[ ] from statistics import mean

    def extract_features(text):

        vocabulary = []
        for idx, word in enumerate(nltk.word_tokenize(text)):
            if not word.isalpha(): continue
            if word in stopwords: continue
            word = word.lower()
            word = lemmatizer.lemmatize(word)
            if word in top_200_spam or top_200_spam:
                vocabulary.append(word)

        fd = nltk.FreqDist(vocabulary)

        return fd
```

```
[ ] spam_comments = dff[dff["categoria"] == "ham"]["coment"].sample(400)
    no_spam_comments = dff[dff["categoria"] == "spam"]["coment"].sample(400)

    features = [
        (extract_features(review), "El comentario es spam")
        for review in spam_comments
    ]
    features.extend([
        (extract_features(review), "El comentario no es spam")
        for review in no_spam_comments
    ])
```

```
[ ] features
```

```
 (FreqDist({'hotel': 11, 'manager': 10, 'show': 6, 'event': 5, 'industry': 5, 'dubai': 4, 'middle': 3, 'east': 3, 'hospitality': 3, 'major': 3, ...}),
  'El comentario no es spam'),
 (FreqDist({'mr': 4, 'bank': 3, 'vincent': 2, 'nnaji': 2, 'standard': 2, 'trust': 2, 'lagos': 2, 'nigeria': 2, 'foreign': 2, 'nigerian': 2, ...}),
  'El comentario no es spam'),
 (FreqDist({'movie': 1, 'censusinternet': 1}), 'El comentario no es spam'),
 (FreqDist({'please': 4, 'confidential': 3, 'write': 3, 'back': 3, 'balakov': 2, 'fund': 2, 'detail': 2, 'via': 2, 'email': 2, 'com': 2, ...}),
  'El comentario no es spam'),
 (FreqDist({'congratulation': 1, 'thanks': 1, 'good': 1, 'friend': 1, 'u': 1, 'xmas': 1, 'prize': 1, 'claim': 1, 'easy': 1, 'call': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'viagra': 2, 'buy': 2, 'hang': 2, 'right': 1, 'medication': 1, 'welcome': 1, 'beginning': 1, 'sexual': 1, 'life': 1, 'using': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'x': 10, 'com': 7, 'messagelabs': 5, 'projecthoneypot': 4, 'spam': 4, 'jul': 4, 'message': 3, 'received': 3, 'squirrelmail': 3, 'email': 3, ...}),
  'El comentario no es spam'),
 (FreqDist({'advertisement': 3, 'see': 2, 'please': 2, 'email': 2, 'would': 2, 'future': 2, 'rollinginthedough': 2, 'info': 2, 'unable': 1, 'graphic': 1, ...}),
```

```
features
```

```
 (FreqDist({'spammer': 5, 'open': 5, 'header': 3, 'relay': 3, 'originating': 2, 'ip': 2, 'proxy': 2, 'used': 2, 'would': 2, 'think': 2, ...}),
  'El comentario es spam'),
 (FreqDist({'enron': 13, 'firm': 5, 'subject': 3, 'success': 3, 'info': 2, 'calger': 2, 'original': 2, 'message': 2, 'sent': 2, 'october': 2, ...}),
  'El comentario es spam'),
 (FreqDist({'information': 4, 'risk': 3, 'management': 3, 'hou': 3, 'ect': 3, 'processing': 3, 'comment': 3, 'kevin': 2, 'moore': 2, 'please': 2, ...}),
  'El comentario es spam'),
 (FreqDist({'anyway': 1, 'seriously': 1, 'hit': 1, 'back': 1, 'otherwise': 1, 'i': 1, 'light': 1, 'armand': 1, 'always': 1, 'shit': 1, ...}),
  'El comentario es spam'),
 (FreqDist({'tunde': 1, 'wishing': 1, 'great': 1, 'day': 1, 'abiola': 1}),
  'El comentario es spam'),
 (FreqDist({'k': 1, 'i': 1, 'head': 1, 'min': 1, 'see': 1}),
  'El comentario es spam'),
 (FreqDist({'enron': 6, 'jean': 5, 'mrha': 4, 'paul': 4, 'please': 4, 'intended': 4, 'recipient': 4, 'forward': 3, 'may': 3, 'week': 3, ...}),
  'El comentario es spam'),
 (FreqDist({'vince': 1, 'would': 1, 'able': 1, 'get': 1, 'copy': 1, 'presentation': 1, 'last': 1, 'night': 1, 'garp': 1, 'coworker': 1, ...}),
  'El comentario es spam'),
 (FreqDist({'office': 3, 'list': 3, 'would': 3, 'know': 2, 'call': 2, 'business': 2, 'enron': 2, 'could': 2, 'thought': 2, 'expense': 2, ...}),
  'El comentario es spam'),
 (FreqDist({'committee': 16, 'prc': 10, 'enron': 9, 'analyst': 8, 'associate': 8, 'program': 7, 'chaired': 6, 'membership': 5, 'follows': 5, 'kevin': 5, ...}),
  'El comentario es spam'),
 (FreqDist({'love': 2, 'know': 1, 'feel': 1, 'make': 1, 'belly': 1, 'warm': 1, 'wish': 1, 'shall': 1, 'meet': 1, 'dream': 1, ...}),
  'El comentario es spam'),
 (FreqDist({'offer': 2, 'permanent': 1, 'fix': 1, 'penis': 1, 'enlargement': 1, 'limited': 1, 'increase': 1, 'atleast': 1, 'inch': 1, 'get': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'supply': 1, 'primo': 1, 'select': 1, 'computer': 1, 'program': 1, 'smallest': 1, 'feasible': 1, 'monetary': 1, 'value': 1, 'company': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'must': 1, 'go': 1, 'like': 1, 'make': 1, 'quotation': 1, 'million': 1, 'race': 1, 'casemardi': 1, 'gras': 1, 'msn': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'r': 4, 'om': 3, 'g': 3, 'opills': 3, 'c': 2, 'l': 2, 'please': 2, 'email': 2, 'note': 2, 'husband': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'la': 4, 'vega': 4, 'high': 3, 'rise': 3, 'boom': 2, 'www': 2, 'verticallv': 2, 'fast': 1, 'becoming': 1, 'major': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'technology': 1, 'expert': 1, 'cruelty': 1, 'like': 1, 'hope': 1, 'spring': 1, 'eternal': 1, 'mortal': 1, 'man': 1, 'moreover': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'one': 4, 'ad': 2, 'l': 2, 'email': 2, 'posted': 1, 'ffa': 1, 'page': 1, 'responded': 1, 'sent': 1, 'e': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'you': 1, 'wo': 1, 'believe': 1, 'true': 1, 'it': 1, 'incredible': 1, 'txts': 1, 'reply': 1, 'g': 1, 'learn': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'regurgitate': 1, 'efficient': 1, 'open': 1, 'turtleback': 1, 'bologna': 1, 'looking': 1, 'medicine': 1, 'obtain': 1, 'pill': 1, 'may': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'cash': 2, 'weekend': 2, 'get': 1, 'lot': 1, 'dear': 1, 'welcome': 1, 'we': 1, 'got': 1, 'biggest': 1, 'best': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'dosage': 10, 'statesman': 5, 'credential': 5, 'biota': 5, 'horoscope': 5, 'banister': 5, 'tampon': 5, 'edit': 5, 'superb': 5, 'biochemic': 5, ...}),
  'El comentario no es spam'),
 (FreqDist({'ebay': 15, 'account': 6, 'u': 5, 'information': 4, 'law': 4, 'user': 3, 'click': 3, 'would': 3, 'like': 3, 'inc': 3, ...}),
  'El comentario no es spam'),
 (FreqDist({'cable': 3, 'ppv': 2, 'filter': 2, 'page': 2, 'good': 1, 'day': 1, 'sir': 1, 'get': 1, 'sport': 1, 'movie': 1, ...}),
  'El comentario no es spam'),
 (FreqDist({'medication': 3, 'need': 2, 'order': 2, 'approved': 2, 'improving': 1, 'quality': 1, 'people': 1, 'life': 1, 'prescription': 1, 'designed': 1, ...}),
  'El comentario no es spam'),
[ ] from random import shuffle

    train_count = len(features)//2
    shuffle(features)
    classifier = nltk.NaiveBayesClassifier.train(features[:train_count])#usamos esta funcion para entrenar el aloritmo de NaiveBayes
    classifier.show_most_informative_features()
    #Un clasificador basado en el algoritmo Naive Bayes. Se utliza para encontrar la probabilidad de una etiqueta en este caso de las palabras
```

```
Most Informative Features
                      cc = 1              El com : El com =     18.0 : 1.0
                     net = 1              El com : El com =     10.0 : 1.0
                     low = 1              El com : El com =      9.4 : 1.0
                    easy = 1              El com : El com =      9.4 : 1.0
                    site = 1              El com : El com =      8.9 : 1.0
                  monday = 1              El com : El com =      8.5 : 1.0
                    ever = 1              El com : El com =      8.1 : 1.0
                    http = 1              El com : El com =      7.8 : 1.0
                  friday = 1              El com : El com =      7.7 : 1.0
                   state = 1              El com : El com =      7.6 : 1.0
```

```
[ ] #Comprovando que tan efectivo es

    nltk.classify.accuracy(classifier, features[train_count:])
```

```
0.8
```

```
[ ] # Prueba con datos que no se han visto

    review = "You need to buy this"
    classifier.classify(extract_features(review))
```

```
'El comentario no es spam'
```

Scikit-Learn Naive Bayes Classifier

```
[ ] from sklearn.naive_bayes import (#Aqu1 se importaron los classifier
        BernoulliNB,
        ComplementNB,
        MultinomialNB
    )
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.neural_network import MLPClassifier
    from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

```
[ ] classifiers = {
        "BernoulliNB": BernoulliNB(),
        "ComplementNB": ComplementNB(),
        "MultinomialNB": MultinomialNB(),
        "KNeighborsClassifier": KNeighborsClassifier(),
        "DecisionTreeClassifier": DecisionTreeClassifier(),
        "RandomForestClassifier": RandomForestClassifier(),
        "LogisticRegression": LogisticRegression(),
        "MLPClassifier": MLPClassifier(max_iter=30000),
        "AdaBoostClassifier": AdaBoostClassifier(),
    }
```

```
[ ] train_count = len(features) // 4
    shuffle(features)

    trained_classifiers = {}

    for name, sklearn_classifier in classifiers.items():
         classifier = nltk.classify.SklearnClassifier(sklearn_classifier)
         classifier.train(features[:train_count])
         accuracy = nltk.classify.accuracy(classifier, features[train_count:])
         trained_classifiers[name] = classifier
         print(F"{accuracy:.2%} - {name}")
```

```
65.33% - BernoulliNB
84.50% - ComplementNB
84.17% - MultinomialNB
56.33% - KNeighborsClassifier
76.17% - DecisionTreeClassifier
79.67% - RandomForestClassifier
84.00% - LogisticRegression
85.83% - MLPClassifier
75.83% - AdaBoostClassifier
```

```
[ ] # Dependiendo de su calificacion escojo el que optenga la mas alta

    import pickle
    f = open('MLPClassifier', 'wb')
    pickle.dump(trained_classifiers["MLPClassifier"], f)
    f.close()
```

```
[ ] # Hago una prueva con el casificador con la notas mas alta

    f = open('MLPClassifier', 'rb')
    deployed_classifier = pickle.load(f)
    f.close()

    #Estima la probabilidad de que ocurra un evento, como votar o no votar,
    #en función de un conjunto de datos determinado de variables independientes.
```

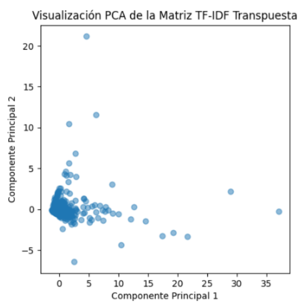
```
[ ] deployed_classifier.classify(extract_features("you need buy this one"))
```

```
'El comentario es spam'
```

Operaciones de Álgebra Matricial con la base de datos utilizadas

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.decomposition import PCA
```

```
[ ]
    # Reemplazar valores NaN (Not a number) con una cadena vacía en la columna 'coment'
    result_df['coment'].fillna('', inplace=True)

    # Aplicación de TF-IDF (Frecuencia de término x Frecuencia inversa de documento)
    tfidf_vectorizer = TfidfVectorizer(max_features=1000)
    X_tfidf = tfidf_vectorizer.fit_transform(result_df['coment'])

    # Cálculo de la transpuesta de los datos TF-IDF
    X_tfidf_transposed = X_tfidf.transpose()

    # Reducción de dimensionalidad con PCA (Reduccion de la dimensionalidad)
    pca = PCA(n_components=2)
    X_tfidf_transposed_reduced = pca.fit_transform(X_tfidf_transposed.toarray())

    # Visualización de la matriz TF-IDF Transpuesta
    plt.figure(figsize=(5, 5))
    plt.scatter(X_tfidf_transposed_reduced[:, 0], X_tfidf_transposed_reduced[:, 1], alpha=0.5)
    plt.title('Visualización PCA de la Matriz TF-IDF Transpuesta')
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.show()
```



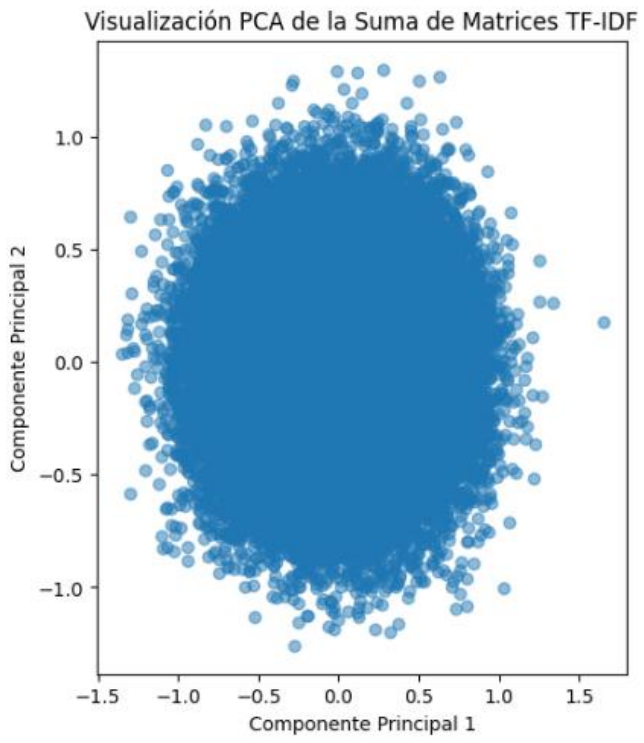
```
[ ] # Obtener las dimensiones de la matriz TF-IDF
    n_rows, n_cols = X_tfidf.shape

    # Crear una matriz de números aleatorios con las mismas dimensiones
    random_matrix = np.random.rand(n_rows, n_cols)

    # Convertir la matriz TF-IDF a formato denso (array)
    tfidf_array = X_tfidf.toarray()

    # Imprimir los datos de las matrices antes de sumar
    print("Matriz TF-IDF (primeros 5 elementos):")
    print(tfidf_array[:5])  # Imprime los primeros 5 elementos para visualización simplificada
    print("\nMatriz Aleatoria (primeros 5 elementos):")
    print(random_matrix[:5])  # Imprime los primeros 5 elementos
```

```
[ ] Matriz TF-IDF (primeros 5 elementos):
    [[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.]]

    Matriz Aleatoria (primeros 5 elementos):
    [[0.12568764 0.11038457 0.83814428 ... 0.21456624 0.76679859 0.17966633]
     [0.71423296 0.75330199 0.85854137 ... 0.04575045 0.39904093 0.90504566]
     [0.91604285 0.94272165 0.30789014 ... 0.36709085 0.08787011 0.86885773]
     [0.24723866 0.2844404  0.54336319 ... 0.57053404 0.41053943 0.18276212]
     [0.80701841 0.96040582 0.3414192  ... 0.65011265 0.92515863 0.02362372]]
```

```
[ ] # Sumar la matriz TF-IDF con la matriz aleatoria
    sum_matrix = tfidf_array + random_matrix

    # Imprimir el resultado de la suma
    print("\nResultado de la Suma (primeros 5 elementos):")
    print(sum_matrix[:5])  # Imprime los primeros 5 elementos del resultado
```

```
    Resultado de la Suma (primeros 5 elementos):
    [[0.12568764 0.11038457 0.83814428 ... 0.21456624 0.76679859 0.17966633]
     [0.71423296 0.75330199 0.85854137 ... 0.04575045 0.39904093 0.90504566]
     [0.91604285 0.94272165 0.30789014 ... 0.36709085 0.08787011 0.86885773]
     [0.24723866 0.2844404  0.54336319 ... 0.57053404 0.41053943 0.18276212]
     [0.80701841 0.96040582 0.3414192  ... 0.65011265 0.92515863 0.02362372]]
```

```
[ ] # Reducción de dimensionalidad con PCA
    pca_sum = PCA(n_components=2)
    sum_matrix_reduced = pca_sum.fit_transform(sum_matrix)

    # Visualización de la suma de matrices
    plt.figure(figsize=(5, 6))
    plt.scatter(sum_matrix_reduced[:, 0], sum_matrix_reduced[:, 1], alpha=0.5)
    plt.title('Visualización PCA de la Suma de Matrices TF-IDF')
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.show()
```


Visualización PCA de la Suma de Matrices TF-IDF
Componente Principal 2
Componente Principal 1