Universidad
Carlos III de Madrid

Universidad Carlos III de Madrid

Métodos Bayesianos, Grado en Estadística y Empresa

Análisis y clasificación de textos

Fabio Scielzo Ortiz

Índice

1 Introducción

En este trabajo se va a realizar un análisis y clasificación de textos. Para ellos se utilizaran dos lenguajes de programación, `Python` y `R`. El trabajo puede dividirse en dos partes bien diferenciadas, una primera parte en la que se trabaja con `Python` y una segunda en la que se usa `R`.

En la primera parte, en la que trabajamos con `Python`, se llevará acabo una descripción y preprocesado del data-set con el que trabajaremos, posteriormente se llevara acabo un análisis de texto, y para finalizar se realizaran tareas de clasificación aplicando algoritmos de clasificación supervisada, especialmente el algoritmo de clasificación ingenua bayesiana.

En la parte en la que trabajamos con `R` se seguirán los pasos del ejemplo ilustrado en clase.

2 Carga de los datos (`Python`)

El data-set con el que vamos a trabajar contiene como observaciones noticias, y como variables la fecha, el título y el texto de la noticia, y si es una noticia falsa (fake new) o es verdadera (no fake new). La variable respuesta será `Fake` . Las variables predictoras que se usaran en el apartado de aplicación de algoritmos de clasificación no aparecen en el data-set original, pero serán creadas usando la información de la variable `texto`

Importamos la libreria `pandas`, que es la liberia de `Python` mas usada para la manipulación y manejo de datos en formato de tabla, es decir, data-frames.

```
import pandas as pd
```

Ahora importamos los datos, que originalmente estan distribuidos en dos data-sets, uno que contiene las fake news (`df_Fake`) y otro que contiene las no fake news (`df_True`):

```
df_Fake = pd.read_csv('Fake.csv')
df_True = pd.read_csv('True.csv')
```

Creamos una variable que indicará en nuestro data-set final si la noticia es fake o no fake:

```
df_Fake['Fake'] = 1
df_True['Fake'] = 0
```

Si para una noticia la nueva variable creada `Fake` toma el valor 1 , indica que es fake new, y si toma el 0 indica que no es fake new.

Ahora concatenamos (por filas) los dos data-sets anteriores, para generar el data-set con el que trabajaremos:

```
Fake_News_Data = pd.concat([df_Fake, df_True])
```

Seleccionamos las columnas (variables) de nuestro interés:

```
Fake_News_Data = Fake_News_Data.loc[: , ['Fake', 'title', 'text', 'date']
↪ ]
```

Añadimos un índice al data-set:

```
Fake_News_Data.index = range(0 , len(Fake_News_Data))
```

Ahora vamos a ver de qué tipo son nuestras variables en `Python` :

```
Fake_News_Data.dtypes
```

```
Fake      int64
title    object
text     object
date     object
dtype: object
```

El tipo `object` es propio de variables no cuantitativos, como categoricas o texto, y el tipo `int64` es propio de variables enteras.

En este caso dejaremos los types como están, salvo el de la variable `Fake` que es categorica y por tanto es más adecuado que su type sea `object`

```
Fake_News_Data['Fake'] = Fake_News_Data['Fake'].astype('object')
```

Calculamos el numero de valores faltantes (NA) en cada una de las variables:

```
Fake_News_Data.isnull().sum()
```

```
Fake     0
title    0
text     0
date     0
```

3 Descripción estadistica de los datos (`Python`)

Hacemos una breve descripción estadistica de las variables del data-set:

```
Fake_News_Data.describe(include='all')
```

```
              Fake                title
count         44898               44898
unique        2                   38729
top           1                   Factbox: Trump fills top jobs for his administ...
freq          23481               14
```

```
              date                  text
count         44898                 44898
unique        2397                  38646
top           December 20, 2017     (no se muestra por tamaño excesivo)
freq          182                   627
```

Esta tabla nos da alguna informacion relevante, como que en el data-set hay mas fake news que no fake news. Concretamente hay 44898 noticias, de las cuales 23481 son fakes y 44898-23481 = 21417 son no fakes.

Vamos ahora a realizar un análisis descriptivo del data-set algo más profundo.

3.1 Gráfico de barras de la variable respuesta (Fake)

Importamos algunas librerias necesarias para realizar este análisis en `Python`

Concretamente la libreria `numpy` da soporte para crear vectores y matrices grandes multi-dimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas. En general es una de las librerias de `Python` más empleadas junto con `pandas`

Tambien importamos las librerias `seaborn` y `matplotlib`que son muy empleadas para visualización de datos (creación de gráficos).

```
import numpy as np

import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt

sns.set(rc={'figure.figsize':(8,8)})
```

Vamos a calcular un gráfico de barras para la variable `Fake`:

```
prop_Fake_yes = len( Fake_News_Data.loc[ Fake_News_Data['Fake']== 1 , :] )
↪  / len(Fake_News_Data)

prop_Fake_no = len( Fake_News_Data.loc[ Fake_News_Data['Fake']== 0 , :] )
↪  / len(Fake_News_Data)
```

```
Fake_News_Data['proportion_Fakes'] = 0


for i in range(0, len(Fake_News_Data)):

    if Fake_News_Data['Fake'][i] == 1 :

        Fake_News_Data['proportion_Fakes'][i] = prop_Fake_yes

    else :

        Fake_News_Data['proportion_Fakes'][i] = prop_Fake_no
```

```
p1 = sns.barplot(x='Fake', y='proportion_Fakes', data=Fake_News_Data,
↪  palette="Spectral")
p1.set_yticks( np.arange(0, 0.85, 0.1)  )
p1.set_xticklabels(['No', 'Yes'])
p1.axes.set(xlabel='Fakes', ylabel='proportion')
```

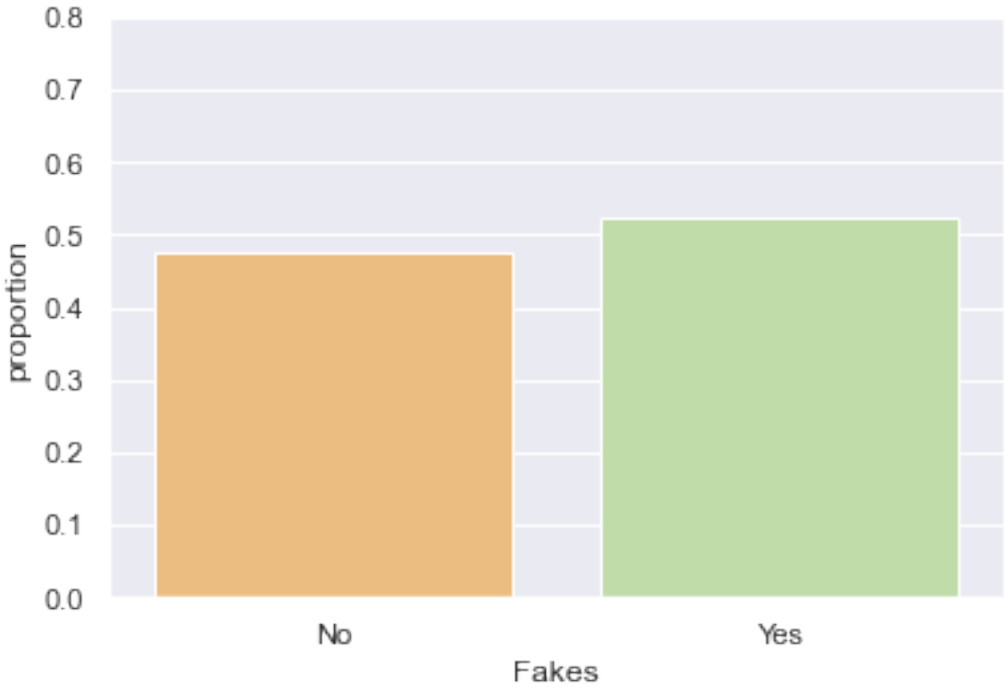

Figure 1: Gráfico de barras de la variable `Fake`

Las proporciones exactas de fake y no fake news son:

```
[prop_Fake_no , prop_Fake_yes]
```

```
[0.47701456635039424, 0.5229854336496058]
```

El número exacto de fake y no fake news es:

```
[prop_Fake_no*len(Fake_News_Data) , prop_Fake_yes*len(Fake_News_Data)]
```

```
[21417.0, 23481.0]
```

Eliminamos la columna `proportion_Fakes` del data-set, que ha sido creada solamente de manera auxiliar para poder generar el gráfico de barras anterior:

```
Fake_News_Data = Fake_News_Data.loc[ : , Fake_News_Data.columns !=
↪  'proportion_Fakes']
```

3.2 Número de palabras por noticia

Una forma de calcular en `Python` el número de palabras de cada notica es la siguiente:

```
Fake_News_Data['word_count'] =
↪  Fake_News_Data['text'].str.split().str.len()
```

Vamos a ver el data-set con la nueva columna `word_count` que contiene el nº de palabras por noticia

```
Fake_News_Data
```

```
      Fake   title

0        1   Donald Trump Sends Out Embarrassing New Year'...
1        1   Drunk Bragging Trump Staffer Started Russian ...
2        1   Sheriff David Clarke Becomes An Internet Joke...
3        1   Trump Is So Obsessed He Even Has Obama's Name...
4        1   Pope Francis Just Called Out Donald Trump Dur...

...    ...                                                ...

44893    0  'Fully committed' NATO backs new U.S. approach...
44894    0  LexisNexis withdrew two products from Chinese ...
44895    0  Minsk cultural hub becomes haven from authorities
44896    0  Vatican upbeat on possibility of Pope Francis ...
44897    0  Indonesia to buy $1.14 billion worth of Russia...
```

```
       text                                                date

0      Donald Trump just couldn t wish all Americans ...  December 31, 2017
1      House Intelligence Committee Chairman Devin Nu...  December 31, 2017
2      On Friday, it was revealed that former Milwauk...  December 30, 2017
3      On Christmas day, Donald Trump announced that ...  December 29, 2017
4      Pope Francis used his annual Christmas Day mes...  December 25, 2017

...                                                  ...                ...

44893  BRUSSELS (Reuters) - NATO allies on Tuesday we...   August 22, 2017
44894  LONDON (Reuters) - LexisNexis, a provider of l...   August 22, 2017
44895  MINSK (Reuters) - In the shadow of disused Sov...   August 22, 2017
44896  MOSCOW (Reuters) - Vatican Secretary of State ...   August 22, 2017
44897  JAKARTA (Reuters) - Indonesia will buy 11 Sukh...   August 22, 2017
```

```
       word_count

0             495
1             305
2             580
3             444
4             420

...           ...

44893         466
44894         125
44895         320
44896         205
44897         210
```

3.3 Numero medio de palabras por noticia en función de si son fake o no

Calculamos ahora la media de palabras de las fakes news y de la sno fake news. Es decir, el nº medio de palabras en el cojuntos de las noticias fake, y por otro lado en el conjutno de las no fake:

```
Fake_News_Data.groupby('Fake')['word_count'].mean()
```

```
Fake   Mean word_count

0        385.640099

1        423.197905
```

4 Preprocesado de texto

En este apartado se vana a hacer una serie de operaciones orientadas al preprocesado de texto, para poder posteriormente realizar analasis mas profundos, y para poder implementar algoritmos de clasificación sobre texto.

Este tipo de preprocesado es básico y fundamental en areas de la ciencia de datos que trabajan con texto, como son la mineria de texto (text minning), el procesamiento del lenguaje natural (PLN) y la recuperación de información (information retrival).

Una de las operaciones centrales del preproceso de textos es la `tokenización`.

4.1 Tokenizacion

Existen algunas librerias de `Python` que tienen funciones para realizar operaciones de tokenizacion, como por ejemplo las librerias `sklearn`, `nltk` o `spaCy`

En este caso no usaremos ninguna función de alguna de esas librerias, sino que crearemos nuestra propia función para realizar la tokenización.

Esta función esta totalmente inspirada en la función creada por el cientifico de datos Joaquín Amat Rodrigo, el cual es el creador del excelente blog sobre ciencia de datos Cienciadedatos.net. En este blog Joaquin tiene un articulo sobre analisis de texto en `Python` en el cual se encuentra la función que ahora vamos a presentar. Ademas muchas otras partes de este trabajo estan basadas en dicho articulo, es por ello que s ele hace una especial mención tanto aqui como en el apartado de bibliografia.

La función `limpiar_tokenizar` toma como input texto y devuelve como output un vector de tokens asociado a ese texto, es decir, un vector con las cadenas caracteres del texto, pero no con cualquier tipo, sino que la función no considera signos de puntuación , palabras que empiezan por “http”, números, espacios en blancos múltiples, tokens con longitud menor que 2.

Un token aqui es considerado como una cadena de caracteres, es decir, una concatenacion de símbolos (sin considerar el espacio en blanco como un símbolo).

Veamos un ejemplo de lo que consideramos tokens:

Dado el siguiente texto:

” Esto es 1 ejemplo de l’limpieza de6 TEXTO https://t.co/rnHPgyhx4Z @cienciadedatos #textmining ”

Los tokens (en sentido estricto, no en el sentido restrictivo que considera la función `limpiar_tokenizar`) asociados a dicho texto son:

[Esto , es , 1 , ejemplo , de , l'limpieza , de6 , TEXTO , https://t.co/rnHPgyhx4Z , @cienciadedatos , #textmining]

```
def limpiar_tokenizar(texto):

    import re

    '''
    Esta función limpia y tokeniza el texto en palabras individuales.
    El orden en el que se va limpiando el texto no es arbitrario.
    El listado de signos de puntuación se ha obtenido de:
    print(string.punctuation)
    y re.escape(string.punctuation)
    '''

    # Se convierte todo el texto a minúsculas:

    nuevo_texto = texto.lower()


    # Eliminacion de paginas web (palabras que empiezan por "http"):

    ## Las cadenas de caracteres que sean enlaces a webs no serán
    ↪  consideradas como tokens

    nuevo_texto = re.sub('http\S+', ' ', nuevo_texto)


    # Eliminacion de signos de puntuación:

    ## Si una cadena de caractrer contiene un signo de puntuacion estos
    ↪  serán eliminados y sustituidos por un   espacio en blanco. Si  por
    ↪  ejemplo tenemos las cadenas  '@FabioScielzo' y 'Fabio@Scielzo' ,
    ## la funcion las transforma en  'FabioScielzo' en el primer caso  y
    ↪  en el par de cadenas 'Fabio' , 'Scielzo' en el segundo. Y si
    ↪  tenemos
    ## una cadena de signos d puntuacion   como '@#!' la elimina
    ↪  directamente.

    regex =
↪   '[\\!\\"\\#\\$\\%\\&\\'\\(\\)\\*\\+\\,\\-\\.\\/\\:\\;\\<\\=\\>\\?\\@\\[\\\\\\]\\^_\\

    nuevo_texto = re.sub(regex , ' ', nuevo_texto)


    # Eliminacion de numeros:

    ## Si una cadena de caracter tiene numeros estos serán eliminados y
    ↪  sustituidos por un espacio en blanco. Si por ejemplo tenemos las
    ↪  cadenas '4FabioScielzo' y 'Fabio44Scielzo' la funcion las
    ↪  transforma en 'FabioScielzo' y 'Fabio' , 'Scielzo' ,
    ↪  respectivamente. Ademas si una cadena solo contienen numeros, por
    ↪  ejemplo '123' la elimina directamente.
```

```
    nuevo_texto = re.sub("\d+", ' ', nuevo_texto)



    # Eliminacion de espacios en blanco multiples:

    ## Si tenemos en un texto dos o mas espacios en blanco consecutivos la
    ↪  funcion los transforma en un solo espacio en blanco. Por ejemplo
    ↪  si tenemos el texto "Fabio     es abogado" la funcion lo
    ↪  transforma en "Fabio es abogado".

    nuevo_texto = re.sub("\\s+", ' ', nuevo_texto)


    # Una vez que a un texto se le han aplicado las operaciones anteriores
    ↪  ya solo quede considerar las cadenas de caracteres de ese texto
    ↪  como tokens, ya que son cadenas con buenas propiedades, a saber,
    ↪  sin signos de puntuacion, sin numeros, sin links de web. Ademas la
    ↪  eliminacion de espacios en blanco multiples es fundamental para
    ↪  que la siguiente operacion funcione bien, ya que en el texto final
    ↪  resultante todas las cadenas estan separadas entre si por un solo
    ↪  espacio, y la siguiente operacion utiliza esa propiedad para
    ↪  identificar a las cadenas, que ya serán considerados tokens en
    ↪  sentido estricto.

    # Obtención de tokens:

    nuevo_texto = nuevo_texto.split(sep = ' ')


    # Eliminacion de tokens con una longitud menor que  2:

    ## Una ultima operacion es solo considerar los tokens obteenidos tras
    ↪  las operaciones anteriores que tengan un tamaño (nº de caracteres)
    ↪  igual o superior a 2 , es decir, dejar fuera tokens con solo un
    ↪  caracter.

    nuevo_texto = [token for token in nuevo_texto if len(token) >=  2]

    return(nuevo_texto)
```

Probamos el funcionamiento de la función `limpiar_tokenizar` con el mismo texto que fue usado antes como ejemplo ilustrativo.

```
test = "Esto es 1 ejemplo de l'limpieza de6 TEXTO  https://t.co/rnHPgyhx4Z
↪  @cienciadedatos #textmining"

print(limpiar_tokenizar(texto=test))
```

```
['esto', 'es', 'ejemplo', 'de', 'limpieza', 'de', 'texto', 'cienciadedatos', 'textmining']
```

Ahora probamos la función `limpiar_tokenizar` con la primera noticia del data-set `Fake_News_Data`:

```
Fake_News_Data['text'][0]
```

```
'Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Ins
```

```
print(limpiar_tokenizar(texto=Fake_News_Data['text'][0]))
```

```
['donald', 'trump', 'just', 'couldn', 'wish', 'all', 'americans', 'happy', 'new', 'year',
```

Ahora aplicamos la función `limpiar_tokenizar` a cada una de las noticias del data-set `Fake_News_Data`

```
Fake_News_Data['text_tokenizado'] = Fake_News_Data['text'].apply(
↪  limpiar_tokenizar )
```

Creamos una columna que identifique las noticias:

```
Fake_News_Data['id_text'] = range(0, len(Fake_News_Data))
```

Vemos como queda tras estos cambios el data-set `Fake_News_Data`:

```
Fake_News_Data
```

	Fake	title	text	date
0	1	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	December 31, 2017
1	1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	December 31, 2017
2	1	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	December 30, 2017
3	1	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	December 29, 2017
4	1	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	December 25, 2017
...	...	...		
44893	0	'Fully committed' NATO backs new U.S. approach...		
44894	0	LexisNexis withdrew two products from Chinese ...		
44895	0	Minsk cultural hub becomes haven from authorities		
44896	0	Vatican upbeat on possibility of Pope Francis ...		
44897	0	Indonesia to buy $1.14 billion worth of Russia...		

```
...                    ...                                          ...

44893  BRUSSELS (Reuters) - NATO allies on Tuesday we...   August 22, 2017
44894  LONDON (Reuters) - LexisNexis, a provider of l...   August 22, 2017
44895  MINSK (Reuters) - In the shadow of disused Sov...   August 22, 2017
44896  MOSCOW (Reuters) - Vatican Secretary of State ...   August 22, 2017
44897  JAKARTA (Reuters) - Indonesia will buy 11 Sukh...   August 22, 2017
```

```
       word_count              text_tokenizado                        id_text

0         495     [donald, trump, just, couldn, wish, all, ameri...      0
1         305     [house, intelligence, committee, chairman, dev...      1
2         580     [on, friday, it, was, revealed, that, former, ...      2
3         444     [on, christmas, day, donald, trump, announced,...      3
4         420     [pope, francis, used, his, annual, christmas, ...      4

...       ...                          ...                            ...

44893     466     [brussels, reuters, nato, allies, on, tuesday,...    44893
44894     125     [london, reuters, lexisnexis, provider, of, le...    44894
44895     320     [minsk, reuters, in, the, shadow, of, disused,...    44895
44896     205     [moscow, reuters, vatican, secretary, of, stat...    44896
44897     210     [jakarta, reuters, indonesia, will, buy, sukho...    44897
```

Creamos un nuevo data-frame solo con las columnas (variables) `id_text` , `text_tokenizado` y `Fake`, en ell que la columna `text_tokenizado` esta expandida, es decir, al ser una columna cuyos elementos son vectores, lo que se hace con la operacion `explode` es expandir cada uno de esos vectores en un nuevo data-frame, es decir, para cada uno de esos vectores se crean tantas filas en el nuevo data-frame como elementos hay en el vector, y en cada una de esas filas la columna `text_tokenizado` contendra un elemento del vector expandido. Visualmente es mas facil de entenderlo como se verá a continuación:

```
Fake_News_Tokens = Fake_News_Data.loc[:, ['id_text', 'text_tokenizado',
↪  'Fake'] ].explode(column='text_tokenizado')


# Renombramos la columna `text_tokenizado` como `token` :

Fake_News_Tokens =
↪  Fake_News_Tokens.rename(columns={'text_tokenizado':'token'})
```

Imprimimos el nuevo data-frame creado `Fake_News_Tokens` al expandir la columna `text_tokenizado` del data-frame `Fake_News_Data`

```
Fake_News_Tokens
```

```
       id_text       token    Fake
0            0      donald       1
0            0       trump       1
0            0        just       1
```

```
0            0      couldn      1
0            0        wish      1

...        ...         ...     ...

44897    44897  technology      0
44897    44897         and      0
44897    44897    aviation      0
44897    44897       among      0
44897    44897      others      0
```

5 Descripción estadística de los datos tras la tokenización

5.1 Numero de tokens del conjunto de noticias en funcion de si son fake o no

```
# nº de palabras (tokens) en el conjunto de textos clasificados como fake
↪  y en los no fake

Fake_News_Tokens.groupby(by='Fake')['token'].count()
```

```
Fake
0    7891501
1    9611544
Name: token, dtype: int64
```

5.2 Numero de tokens *únicos* del conjunto de noticias en funcion de si son fake o no

```
# nº de palabras (tokens) *unicos* en el conjunto de textos clasificados
↪  como fake y en los no fake

Fake_News_Tokens.groupby(by='Fake')['token'].nunique()
```

```
Fake
0    78020
1    85642
Name: token, dtype: int64
```

5.3 Numero de tokens en cada una de las noticias individualmente

```
# nº de palabras (tokens) en cada texto individual clasificados como fake
↪  y en los no fake

df1 = pd.DataFrame( Fake_News_Tokens.groupby(by = ["id_text" , "Fake"]
↪  )["token"].count().rename('nº_tokens') )
```

```
df1
```

		nº_tokens
id_text	Fake	
0	1	447
1	1	294
2	1	563
3	1	426
4	1	415

```
 ...       ...          ...

44893       0          433
44894       0          120
44895       0          307
44896       0          196
44897       0          197
```

Hay noticias que no tienen tokens :

```
df1.loc[df1['nº_tokens'] == 0, :]
```

```
                nº_tokens
id_text  Fake

9358      1           0
10923     1           0
11041     1           0
11190     1           0
11225     1           0

 ...     ...         ...

21857     1           0
21869     1           0
21870     1           0
21873     1           0
32451     0           0
```

Algunos ejemplos de estas noticias son los siguientes:

```
Fake_News_Data.loc[Fake_News_Data.id_text == 9358]
```

```
        Fake   title
9358     1     https://100percentfedup.com/served-roy-moore-v...


        text
9358    https://100percentfedup.com/served-roy-moore-v...


        date  word_count
9358              1


      text_tokenizado   id_text
9358        []           9358
```

```
Fake_News_Data.loc[Fake_News_Data.id_text == 10923]
```

```
       Fake      title
10923    1       TAKE OUR POLL: Who Do You Think President Trum...

       text     date            word_count   text_tokenizado   id_text
10923           May 10, 2017        0              []           10923
```

Nos quedamos por tanto solo con las noticias que tienen algun token :

```
df2 = df1.loc[df1['nº_tokens'] != 0, :]

df2
```

```
                nº_tokens
id_text   Fake

0          1          447
1          1          294
2          1          563
3          1          426
4          1          415

...       ...         ...

44893      0          433
44894      0          120
44895      0          307
44896      0          196
44897      0          197
```

Calculamos el numero medio de tokens para las noticas que tienen uno o mas tokens en funcion se si son fake o no:

```
df2.groupby("Fake")["nº_tokens"].agg(['mean'])
```

```
            mean
Fake
0     368.486225
1     422.169983
```

Se puede interpretar como la longitud media de las noticas fake y de las no fake

Hay diferencias entre lo obtenido mediante esta operación y lo obtenido al usar el siguiente código, que fue visto anteriormente:

```
Fake_News_Data['word_count'] =
↪  Fake_News_Data['text'].str.split().str.len()

Fake_News_Data.groupby('Fake')['word_count'].mean()
```

```
Fake   Mean word_count

0        385.640099

1        423.197905
```

Y esto es debido a que el código `Fake_News_Data['text'].str.split()` hace una operacion similar a la realizada por nuestra funcion `limpiar_tokenizar` pero **no exactamente igual**, y esto lleva a que con la primera opcion se obtiene un conjunto de tokens diferente al obtenido con la funcion `limpiar_tokenizar`, en los distintos documentos, y esto lleva a que la longitud de los documentos sea diferente si se consideran los tokens obtenidos con `Fake_News_Data['text'].str.split()` a si se usan los obtenidos con `limpiar_tokenizar`, llo que lleva a diferencias en las longitudes medias obtenidas.

```
Fake_News_Data['text'].str.split()
```

```
0        [Donald, Trump, just, couldn, t, wish, all, Am...
1        [House, Intelligence, Committee, Chairman, Dev...
2        [On, Friday,, it, was, revealed, that, former,...
3        [On, Christmas, day,, Donald, Trump, announced...
4        [Pope, Francis, used, his, annual, Christmas, ...
                              ...
44893    [BRUSSELS, (Reuters), -, NATO, allies, on, Tue...
44894    [LONDON, (Reuters), -, LexisNexis,, a, provide...
44895    [MINSK, (Reuters), -, In, the, shadow, of, dis...
44896    [MOSCOW, (Reuters), -, Vatican, Secretary, of,...
44897    [JAKARTA, (Reuters), -, Indonesia, will, buy, ...
```

Como se pueden ver con el código anterior se obtiene por ejemplo que '-' y ', Donald' son tokens , cuando con la función `limpiar_tokenizar` no serían considerados un tokens.

Otra forma de calcular lo anterior:

```
m0 = (
↪  Fake_News_Tokens.loc[Fake_News_Tokens['Fake']==0].groupby('id_text')['token'].count()
↪  ).mean()
```

```
m1 = (
↪  Fake_News_Tokens.loc[Fake_News_Tokens['Fake']==1].groupby('id_text')['token'].count()
↪  ).mean()
```

```
pd.DataFrame({'fake_new': [0,1] , 'tokens_mean':[m0 , m1]})
```

```
   fake_new  tokens_mean
0         0   368.469020
1         1   409.332822
```

```
df = pd.DataFrame(  (Fake_News_Tokens.groupby(by = ["Fake", "token"]
↪  )["token"].count().unstack(fill_value=0).stack().reset_index(name='frecuencia_token')

# .unstack(fill_value=0).stack() para que tambien aparezcan los tokens con
↪  count = 0 , si no solo aprecerian los que tienen count > 0.
```

```
df # Nos da el nº de veces que sale cada token en el conjunto de las
↪  noticas fake y por otro lado en el de las no fake (solo salen tokens
↪  con count > 0 )
```

	Fake	token	frecuencia_token
0	0	aa	22
1	0	aaa	7
2	0	aaaaaaaand	0
3	0	aaaaackkk	0
4	0	aaaaapkfhk	0
...	...	...	...
251605	1	""it	0
251606	1	""when	0
251607	1	•if	0
251608	1	$\surd$	0
251609	1	$\Rightarrow$	0

```
df.loc[df['token']=='yes' , ] # El token 'yes' aprece 1775 veces en el
↪  conjunto de las fake news y 336 en el de las no fake news
```

Fake

token

frecuencia_token

116577

0

yes

336

242382

1

yes

1775

```
df.loc[df['token']=='true' , ] # El token 'true' aparece 2595 veces en el
↪  conjunto de las fake news y 412 en el de las no fake news
```

Fake

token

frecuencia_token

106608

0

true

412

232413

1

true

2595

```
df.loc[df['Fake']==0 , ] # frecuencia de tokens en el conjunto de las no
↪  fake news
```

Fake

token

frecuencia_token

0

0

aa

22

1

0

aaa

7

2

0

aaaaaaaand

0

3

0

aaaaackkk

0

4

0

aaaaapkfhk

0

. . .

. . .

. . .

. . .

125800

0

” ”it

1

125801

0

” ”when

1

125802

0

• if

3

125803

0

√

3

125804

0

⇒

1

125805 rows × 3 columns

```
df.loc[df['Fake']==1 , ] # nº de tokens en el conjunto de las fake news
```

Fake

token

frecuencia_token

125805

1

aa

24

125806

1

aaa

9

125807

1

aaaaaaaand

1

125808

1

aaaaackkk

1

125809

1

aaaaapkfhk

1

. . .

. . .

. . .

. . .

251605

1

” ”it

0

251606

1

” ”when

0

251607

1

• if

0

251608

1

√

0

251609

1

⇒

0

125805 rows × 3 columns