Preparación de los datos para análisis y Machine Learning

En este notebook crearemos un *pipeline* de preprocesamiento de texto similar al visto al principio del estudio, pero más avanzado y haciendo uso de librerías como *spaCy* y *textacy*. Una vez completado, se obtendrá un texto limpio y tokenizado listo para su análisis.

Para este caso, se va a hacer uso del dataset creado en el apartado anterior, con más de 2000 comentarios del repoositorio *zigbee2mqtt*.

Al igual que en los cuadernos anteriores, comenzaremos cargando unos ajustes predefinidos para la ejecución del entorno virtual de python.

```
import sys, os

#Carga del archivo setup.py
%run -i ../pyenv_settings/setup.py

#Imports y configuraciones de gráficas
%run "$BASE_DIR/pyenv_settings/settings.py"

#Reset del entorno virtual al iniciar la ejecución
#%reset -f

%reload_ext autoreload
%autoreload 2
%config InlineBackend.figure_format = 'png'

# to print output of all statements and not just the last
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# otherwise text between $ signs will be interpreted as formula and
printed in italic
pd.set_option('display.html.use_mathjax', False)
```

```
You are working on a local system.
Files will be searched relative to "..".
```

Carga de los datos en Pandas

Cargaremos el dataset creado anteriormente con todos los comentarios de un repositorio de Github en Pandas, concretamente el archivo .csv (hay dos idénticos, uno en formato .csv y otro en .json)

```
#Ruta del archivo
file_path = "../data/output.csv"

#Carga del archivo en un DataFrame
df = pd.read_csv(file_path)
```

Antes de empezar a trabajar con los datos, revisaremos el nombre de las columnas y se cambiarán por otros nombres más genéricos en caso de considerarse necesario para una mejor coprensión y maniobrabilidad con el documento.

```
print(df.columns)
```

Para el renombramiento de las columnas, definiremos un diccionario *column_mapping* en el que cada entrada corresponderá con el nombre de la columna original y el nuevo que se le dará.

Si se considera que algunas columnas no son necesarias para el análisis, se pueden descartar nombrándolas como *None* o directamente sin incluirlas en el diccionario.

Viendo las columnas con las que cuenta el DataFrame se ve a simple vista que hay algunas columnas irrelevantes para el estudio, como las URLs, node_id, fechas de creación y actualización del post, asociaciones y la columna "performed_via_github_app". Estas serán descartadas a continuación sin incluirlas en el diccionario:

```
import ast #Para convertir cadenas JSON en objetos Python para la
eliminación de campos innecesarios del campo 'user'

column_mapping = {
    'id' : 'id',
    'user' : 'user',
    'body' : 'text',
    'reactions' : None,
    'url' : None,
    'html_url' : None,
    'issue_url' : None,
    'node_id' : None,
    'created_at' : None,
    'update_at' : None,
    'author_association' : None,
    'performed_via_github_app' : None
}

#Se definen las columnas que se mantendrán
columns = [c for c in column_mapping.keys() if column_mapping[c] !=
None]

#Seleccionar y renombrar las columnas
df = df[columns].rename(columns=column_mapping)

#Normalizamos la columna user para extraer únicamente la información
```

```
que interesa
# user_data = pd.json_normalize(df['user'])

# #Asegurar que las columnas que nos interesan existen
# if 'login' in user_data.columns and 'id' in user_data.columns:
#     df[['login', 'id']] = user_data[['login', 'id']]
# else:
#     raise ValueError("Las columnas 'login' e 'id' no se encuentran
en los datos normalizados de 'user'")

# #Se elimina la columna 'user' original
# df = df.drop(columns=['user'])

#Muestra de una entrada para comprobar que se ha ejecutado
correctamente
df.sample(1).T
```

```
878
id
1584971222
user  {'login': 'davix10', 'id': 6436570, 'node_id':
'MDQ6VXNlcjY0MzY1NzA=', 'avatar_url': 'https://private-
avatars.githubusercontent.com/u/6436570?
jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJna...
text
Were you able to solve the problem?
```

Guardado y carga de un Data Frame

Para guardar el Data Frame en disco se hará uso de una base de datos SQL utilizando SQLite. No es necesario contar con conocimientos avanzados de SQL pues se hará uso de la librería *sqlite3* de python que integra todas las funciones necesarias para trabajar con este tipo de bases de datos, como mucho, se usarán sentencias SQL básicas para realizar acciones sobre la base de datos..

Cuando guardamos el Data Frame en la base de datos, no se almacena el índice del Data Frame, y todos los datos ya existentes son sobreescritos.

```
#sqlite3 ya está importado en el archivo settings.py cargado al
iniciar el programa

#Damos nombre a la DB, nos conectamos a ella, guardamos los datos, y
se cierra conexión
db_name = "../data/zigbee2mqtt_comments.db"
con = sqlite3.connect(db_name)
df.to_sql("comments", con, index=False, if_exists="replace")
con.close()
```

```
2678
```

El Data Frame se lee de forma muy sencilla:

```
con = sqlite3.connect(db_name)
df = pd.read_sql("select * from comments", con)
```

Limpieza de los datos

Antes de la tokenización de los datos es necesario limpiar los datos recopilados de ruido innecesario y distintos formatos en el texto. Algunos ejemplos de estos pueden ser los caracteres especiales, URLs incluidas en los comentarios, etiquetas, emoticonos, etc.

Para esta función se usarán expresiones regulares junto con la librería *regex* para detectar y eliminar todos estos elementos innecesarios para el posterior análisis.

```
#import re -> importado en settings.py

RE_SUSPICIOUS = re.compile(r'[&#<>{}\[\]\\]') #símbolos sospechosos de introducir ruido

def impurity(text, min_len=10): #se ignoran textos de menos de 10 caracteres
    if text == None or len(text) < min_len:
        return 0
    else:
        return len(RE_SUSPICIOUS.findall(text))/len(text)
```

Ahora se procederá a depurar y eliminar el ruido en los comentarios de los posts extraídos del repositorio con el que hemos estado trabajando hasta este momento,

```
#Se añade la columna "impurity" al Data Frame que mostrará el porcentaje de cada comentario
df['impurity'] = df['text'].progress_apply(impurity, min_len=10)

#Algunas muestras de los registros con más ruido
df[['text', 'impurity']].sort_values(by='impurity', ascending=False).head(5)
```

```
100%|██████████| 2678/2678 [00:00<00:00, 125179.38it/s]


text  \
1877  > If I put this config in my zigbee2mqtt 1.25.0-1 and also the below in the config folder it doesn't work. \n> \n> \n> \n> data_path: /config/zigbee2mqtt\n> \n> socat:\n> \n>   enabled: false\n> \...
1902
Sure, done in #307.
1866  > > Thanks mate. Gonna check this tomorrow in the afternoon to (hopefully) help you out. I'm now gonna take a nap.\n> \n> \n> \n> Mate i found the problem.\n> \n> if you reset to default values, t...
```

```
1884  > It seems to me that within the configuration tab of the
'Zigbee2MQTT' addon the following lines need to be present (and the
rest removed):\n> \n> ```\n> \n> data_path: /config/zigbee2mqtt\n> \
n>...
1446                                                That works and shows
correct state in HA.\r\n\r\nJust FYI - I forgot to mention before that
`{{ value_json.tilt }}` needs to be "{{ value_json.tilt }}" to work.


      impurity
1877      0.08
1902      0.05
1866      0.05
1884      0.05
1446      0.05
```

Como se observa, el grado de impurezas no es demasiado elevado, pero si se omiten facilitará el trabajo el análisis, ademaś de que siempre se tiene que tener en cuenta debido a que se está trabajando con contenido generado por usuarios, y este puede ser un caso excepcional en el que no hay demasiado ruido.

Conteo de otras posibles palabras que pueden introducir ruido

Se importará la función *count_words* utilizada en *1-textual_data* para realizar el conteo de palabras de otras etiquetas que no se han tenido en cuenta y que también pueden introducir ruido.

```
from collections import Counter

def count_words(df, column='tokens', preprocess=None, min_freq=2):

    #procesa los tokens y actualiza el contador
    def update(doc):
        tokens = doc if preprocess is None else preprocess(doc)
        counter.update(tokens)

    #crea el contador y recorre todos los datos
    counter = Counter()
    df[column].progress_map(update)

    #transforma el contador a dataframe
    freq_df = pd.DataFrame.from_dict(counter, orient='index',
columns=['freq'])
    freq_df = freq_df.query('freq >= @min_freq')
    freq_df.index.name = 'token'

    return freq_df.sort_values('freq', ascending=False)

count_words(df, column='text', preprocess=lambda t: re.findall(r'<[\
w/]*>', t))
```

```
100%|██████████| 2678/2678 [00:00<00:00, 168201.77it/s]
```

token	freq
<anonymous>	51
<redacted>	5
</details>	5
<details>	5
</summary>	5
<summary>	5
<REDACTED>	4
<template>	4
</script>	4
<snip>	4

Eliminación de ruido con Expresiones Regulares

Se va a crear una función que definirá una serie de expresiones regulares que serán utilizadas para detectar en el texto una serie de patrones que cumplen aquellas palabras susceptibles de introducir ruido. Estas serán sustituidas por texto plano o eliminadas directamente del texto.

```
import html

def clean(text):
    # convert html escapes like &amp; to characters.
    text = html.unescape(text)
    # tags like <tab>
    text = re.sub(r'<[^<>]*>', ' ', text)
    # markdown URLs like [Some text](https://....)
    text = re.sub(r'\[([^\[\]]*)\]\([^\(\)]*\)', r'\1', text)
    # text or code in brackets like [0]
    text = re.sub(r'\[[^\[\]]*\]', ' ', text)
    # standalone sequences of specials, matches &# but not #cool
    text = re.sub(r'(?:^|\s)[&#<>{}\[\]+|\\:-]{1,}(?:\s|$)', ' ',
text)
    # standalone sequences of hyphens like --- or ==
    text = re.sub(r'(?:^|\s)[\-=\+]{2,}(?:\s|$)', ' ', text)
    # sequences of white spaces
    text = re.sub(r'\s+', ' ', text)

    return text.strip()
```

Ahora se aplicará esta función a la columna "text" del Data Frame que almacena los comentarios de los usuarios en el repositorio, además, se añadirá una nueva columna con el texto limpio, de modo que se pueda visualizar más fácilmente los cambios entre el texto original y el texto sin ruido.

```
df['clean_text'] = df['text'].progress_apply(clean)

#Muestras de la columna "clean_text"
print(df[['text', 'clean_text']].sample(5))
```

Ya se había indicado antes que con suerte la información extraída no contenía demasiado ruido, pese a eso, se puede ver a simple vista la diferencia entre algunos textos originales y los textos ya sin ruido.

Normalización de caracteres con *textacy*

Caracteres especiales como los acentos, apóstrofes, diéresis, etc. pueden ser un problema a la hora de tokenizar un texto, por ello se normalizará sustituyendo estos caracteres por equivalentes ASCII para evitar así inconvenientes.

Se utilizará la librería *textacy* creada para trabajar junto con *spaCy*.

Enmascaramiento de datos basados en patrones

Del mismo modo, hay patrones como URLs y correos electrónicos que habitualmente tampoco serán de ayuda para el análisis de la información, es por ello que estos patrones se pueden sustituir/enmascarar por un texto simple en lugar de eliminarlo del texto porque cabe la posibilidad de perder el contexto de la frase

```
import textacy
import textacy.preprocessing as tprep

# En caso de que se cuente con una versión menor a la 0.11
if textacy.__version__ < '0.11':
    def normalize(text):
        text = tprep.normalize_hyphenated_words(text)
        text = tprep.normalize_quotation_marks(text)
        text = tprep.normaliza_unicode(text)
        text = tprep.remove_accents(text)

        return text

else:
    #En mi caso cuento con la versión 0.13
    def normalize(text):
        text = tprep.normalize.hyphenated_words(text)
        text = tprep.normalize.quotation_marks(text)
        text = tprep.normalize.unicode(text)
        text = tprep.remove.accents(text)
        #Enmascaramiento de patrones
        text = tprep.replace.urls(text)
        text = tprep.replace.emails(text)
        text = tprep.replace.emojis(text)
```

```
        return text
```

En este caso, se aplicará la normalización sobre el texto limpio sin ruido obtenido en el apartado anterior.

```
df['normalized_text'] = df['clean_text'].progress_apply(normalize)

#Muestras de la columna "clean_text"
print(df[['text', 'clean_text', 'normalized_text']].sample(5))
```

Tras haber obtenido un texto limpio, se realiza la conexión a la base de datos para guardar los cambios realizados.

```
con = sqlite3.connect(db_name)
df.to_sql("comments", con, index=False, if_exists="replace")
con.close()
```

```
2678
```

Tokenización

Como ya se explicó en el primer capítulo del estudio, vamos a tokenizar la información que tenemos para facilitar así su análisis.

En este apartado se van a ver dos versiones distintas, una utilizando *expresiones regulares* (similar al visto en el primer capítulo) y otra con la librería *NLTK*.

Como lo que se va a tokenizar no es un texto simple, si no todas las entradas de la columna *"normalized_text"*, se creará una función que reciba como parámetro un texto que corresponderá con cada entrada de la columna, lo tokenizará y lo devolverá, para a continuación guardarlo en otra columna llamada *"tokens"*.

Tokenización con Expresiones Regulares

```
#Función tokenizadora
def tokenize(text):
    tokens = re.findall(r'\w\w+', text)

    return tokens

#Tokenización de cada entrada del Data Frame
df['tokens'] = df['normalized_text'].progress_apply(tokenize)

#Impresión por pantalla de algunas muestras de entradas tokenizadas
print(df[['normalized_text', 'tokens']].sample(3))
```

Se se observa que algunas expresiones como caracteres especiales o emojis se han perdido, se puede modificar la función para incluir tipos de expresiones que se desean incluir en la tokenización.

```
RE_TOKEN = re.compile(r"""
               ( [#]?[@\w'’\.\-\:]*\w     # words, hash tags and email
adresses
               | [:;<]\-?[\)\(3]          # coarse pattern for basic
text emojis
               | [\U0001F100-\U0001FFFF]  # coarse code range for
unicode emojis
               )
               """, re.VERBOSE)

def ttokenize(text):
    return RE_TOKEN.findall(text)

df['tokens'] = df['normalized_text'].progress_apply(ttokenize)

print(df[['normalized_text', 'tokens']].sample(3))
```

```
100%|██████████| 2678/2678 [00:00<00:00, 24898.08it/s]


normalized_text  \
1132  Can you provide the herdsman debug log when trying to start? See
_URL_ on how to enable the herdsman debug logging. Note that this is
only logged to STDOUT and not to log files. I have enabled the...
2634  Murada, just put the next few lines to beginning of the new js:
const fz =
require('zigbee-herdsman-converters/converters/fromZigbee'); const tz
= require('zigbee-herdsman-converters/converters/to...
1888  As you could have read in the zigbee1mqtt release notes there
are breaking changes in this update. I'm having a hard time
understanding the steps to be taking to get things up and running
again I ...


tokens
1132  [Can, you, provide, the, herdsman, debug, log, when, trying, to,
start, See, _URL_, on, how, to, enable, the, herdsman, debug, logging,
Note, that, this, is, only, logged, to, STDOUT, and, not, to...
2634  [Murada, just, put, the, next, few, lines, to, beginning, of,
the, new, js, const, fz, require, 'zigbee-herdsman-converters,
converters, fromZigbee, const, tz, require, 'zigbee-herdsman-
converters...
1888  [As, you, could, have, read, in, the, zigbee1mqtt, release,
notes, there, are, breaking, changes, in, this, update, I'm, having,
a, hard, time, understanding, the, steps, to, be, taking, to, get, ...
```

```
#Tras la tokenización, en la columna "tokens" se almacena una lista de palabras (tokens)
#Como sqlite no sabe como manejas valores de tipo lista, hay que transformar estos en texto
df['tokens'] = df['tokens'].apply(lambda x: ','.join(x) if isinstance(x, list) else x)

#Guarda el Data Frame en la base de datos
con = sqlite3.connect(db_name)
df.to_sql("comments", con, index=False, if_exists="replace")
con.close()
```

```
2678
```

Tokenización con biblioteca NLTK

El resultado será similar al obtenido en el apartado anterior (independientemente de la biblioteca utilizada, al fin y al cabo). El usuario es el que decide si desea definir las expresionres regulares por su cuenta o utilizar un diccionario ya aportado por una librería.

Cabe recalcar que hay librerías con diccionarios muy completos, y estos siempre son accesibles para la modificación por parte del usuario, ya sea para añadir o para eliminar expresiones.

En este caso se va a utiliar el módulo *punkt* de NLTK, un paquete que incluye herramientas y datos preentrenados para la tokenización de textos.

```
print(df.columns)

import nltk
from tqdm import tqdm

tqdm.pandas() #Inicializa el soporte para Pandas

nltk.download('punkt')
nltk.download('punkt_tab')

#Función para tokenizar las columnas indicadas del Data Frame con NLTK
def nltk_tokenize(text):
    try:
        tokens = nltk.tokenize.word_tokenize(text)
        return tokens
    except Exception as e:
        print(f"Error al tokenizar: {e}")
        return []

# # Rellenar valores NaN con una cadena vacía antes de tokenizar
df['normalized_text'] = df['normalized_text'].fillna('')

df['tokens'] = df['normalized_text'].progress_apply(nltk_tokenize)
```

```
print(df[['normalized_text', 'tokens']].head())
```

```
[nltk_data] Downloading package punkt to /home/diego/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

True

[nltk_data] Downloading package punkt_tab to /home/diego/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!

True

100%|██████████| 2678/2678 [00:01<00:00, 1490.41it/s]


normalized_text  \
0
This issue is stale because it has been open 30 days with no activity.
Remove stale label or comment or this will be closed in 7 days
1  Also, after updating the z2m, cyclic reboots began ''' Starting
Zigbee2MQTT without watchdog. INFO: Preparing to start... INFO: Socat
not enabled INFO: Starting Zigbee2MQTT... Starting Zigbee2MQTT...
2  Hi ! Since 2 or 3 days, MQTT suddenly fail. A few messages in the
log, many auto restart, and works again ... Very strange. In the log
INFO: Preparing to start... ERROR: Got unexpected response fr...
3  I don't know if it's exactly the same, but since v1.42 I have
trouble with Z2M. It restarts x times a day without further notice. I
think it is a software issue, becaue in older versions I didn't ...
4  Hello, I have the same problem, see the log file zigbee2mqtt_2024-
12-17T07-07-00.298Z.log for details. Everything runs normally until
00:37:59, after which no more data is received. At 03:00:06 th...


tokens
0                                        [This, issue, is, stale,
because, it, has, been, open, 30, days, with, no, activity, ., Remove,
stale, label, or, comment, or, this, will, be, closed, in, 7, days]
1  [Also, ,, after, updating, the, z2m, ,, cyclic, reboots, began, ``,
', Starting, Zigbee2MQTT, without, watchdog, ., INFO, :, Preparing,
to, start, ..., INFO, :, Socat, not, enabled, INFO, :, Start...
2  [Hi, !, Since, 2, or, 3, days, ,, MQTT, suddenly, fail, ., A, few,
messages, in, the, log, ,, many, auto, restart, ,, and, works,
again, ..., Very, strange, ., In, the, log, INFO, :, Preparing, to...

3  [I, do, n't, know, if, it, 's, exactly, the, same, ,, but, since,
v1.42, I, have, trouble, with, Z2M, ., It, restarts, x, times, a, day,
without, further, notice, ., I, think, it, is, a, software,...
4  [Hello, ,, I, have, the, same, problem, ,, see, the, log, file,
zigbee2mqtt_2024-12-17T07-07-00.298Z.log, for, details, ., Everything,
runs, normally, until, 00:37:59, ,, after, which, no, more, d...
```