

Métodos no supervisados: Topic Modeling y Clustering

Cuándo se está trabajando con un dataset compuesto de una gran cantidad de documentos se quiere saber cuál es el tema del que tratan. Es muy complicado asignar un único tema al corpus debido a qué está compuesto por textos con gran variedad de temas, más en este caso en el que se trabaja con un dataset conformado por comentarios de usuarios.

Es en este punto cuando entra en juego el Topic Modeling (Modelado de temas). Este tipo de modelado se utiliza para determinar la estructura global del corpus mediante diversas técnicas estadísticas, no para asignar un tema concreto a cada uno de los documentos del corpus.

En este apartado se estudiarán varios métodos para el modelado de temas que permitirán entender su funcionamiento y cómo pueden ser utilizados para generar resúmenes de los documentos de forma rápida.

Para esta tarea se utilizará el dataset conformado por los comentarios de usuarios del repositorio "Zigbee2mqtt" que ya ha sido utilizado en tareas anteriores. El DataFrame en cuestión permitirá agilizar el proceso de modelado de temas porque este ya está tokenizado, vectorizado, etc.

Al igual que en todos los apartados, se comienza importando los ajustes del proyecto junto con la base de datos en la que se encuentra almacenado el Data Frame.

```
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"

%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 "...".
```

Usaremos el DataFrame *posts_nlp* de la base de datos porque los datos almacenados ya están preparados en gran parte para el procesamiento de lenguaje natural (NLP).

```
#Conexión con la base de datos en la que tenemos guardado el Data
Frame
db_name = "../data/zigbee2mqtt_comments.db"
con = sqlite3.connect(db_name)
df = pd.read_sql("select * from posts_nlp", con)
con.close()

#Comprobación de que se ha cargado correctamente
print(df.columns)
#print(df[['normalized_text', 'tokens']].head(4))

Index(['id', 'user', 'text', 'impurity', 'clean_text',
       'normalized_text',
       'tokens', 'lemmas', 'adjs_verbs', 'nouns', 'noun_phrases',
       'adj_noun_phrases', 'entities'],
      dtype='object')
```

Pasos previos

Antes de comenzar el modelado, es recomendable conocer la información del corpus para así determinar cuáles son las entidades que se analizarán.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2678 entries, 0 to 2677
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
 --- 
 0   id                2678 non-null    int64  
 1   user               2678 non-null    object 
 2   text               2678 non-null    object 
 3   impurity           2678 non-null    float64
 4   clean_text         2678 non-null    object 
 5   normalized_text    2678 non-null    object 
 6   tokens              2678 non-null    object 
 7   lemmas              2678 non-null    object 
 8   adjs_verbs          2678 non-null    object 
 9   nouns               2678 non-null    object 
 10  noun_phrases       2678 non-null    object 
 11  adj_noun_phrases  2678 non-null    object 
 12  entities            2678 non-null    object 
dtypes: float64(1), int64(1), object(11)
memory usage: 272.1+ KB
```

A priori parece una buena base para el modelado teniendo en cuenta que no hay elementos nulos en ninguna columna. De todos modos, se puede imprimir por pantalla alguna muestra de estos textos para comprobar si estos contienen caracteres especiales como puede ser la tabulación (\t), nueva línea (\r), retorno de carro (\r), etc.

```
print(repr(df.iloc[0]["normalized_text"])[0:200]))  
print(repr(df.iloc[-1]["normalized_text"])[0:200]))  
  
'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'  
"Is this error? I have (old config) ''' socat: restartdelay: 1  
initialdelay: 1 ''' in config but when i check default config this  
positions will disappear."
```

Como el data frame ya había sido procesado y limpiado previamente, no se encuentran caracteres especiales que puedan dificultar el modelado.

Una vez completado este paso, se pasaría a la vectorización de los datos. En este caso, los datos ya fueron vectorizados anteriormente en *e_featureEngineering_and_syntactSimilarity.ipynb* de este mismo proyecto, permitiendo ahorrar tiempo y agilizar el proceso para cumplir el objetivo de este apartado.

Factorización de matrices no negativas (NMF) con scikit-learn

Conceptualmente, la forma más sencilla de hallar la estructura implícita en el corpus es la factorización de la matriz de términos, pero puede presentar un gasto computacional muy elevado.

En lugar de esto, se puede realizar una factorización aproximada que es menos costoso y al mismo tiempo arroja buenos resultados.

Algunos métodos de álgebra lineal permiten representar la matriz como el producto de otras dos matrices no negativas. En este caso se nombrará la matriz original como *V*, y los factores *W* y *H*.

Creación de Modelos temáticos utilizando NMF

Casi todos los modelados de temas necesitan un número de temas como parámetro de entrada. En lugar de utilizar todos los temas de todos los textos, se utilizará un número aleatorio para esta tarea, por ejemplo 10. Este número es variable en cualquier caso, al fin y al cabo lo que se busca es un resultado más afinado, pero tampoco puede ser un número demasiado elevado de forma que el gasto computacional exceda los valores deseados.

Como la vectorización de los textos fue llevada a cabo en otro documento del proyecto, en este se deberá realizar el proceso otra vez para poder hacer uso de la variable que almacena los vectores.

```

from sklearn.feature_extraction.text import TfidfVectorizer
from spacy.lang.en.stop_words import STOP_WORDS as stopwords

tfidf_text_vectorizer = TfidfVectorizer(stop_words=list(stopwords),
min_df=5, max_df=0.7)
tfidf_text_vectors =
tfidf_text_vectorizer.fit_transform(df['normalized_text'])
tfidf_text_vectors.shape

(2678, 1721)

# Comprobación de que no se pierde información
df['normalized_text'].info()
print(df['normalized_text'].sample(5))

<class 'pandas.core.series.Series'>
RangeIndex: 2678 entries, 0 to 2677
Series name: normalized_text
Non-Null Count Dtype
-----
2678 non-null    object
dtypes: object(1)
memory usage: 21.0+ KB
2170    Hello in attach you see my zig network If i try to add a
router (this one _URL_ like other two that you can see on my actual
map (identical in hardware and firmware) behind a indicated router
(red...
842
Should also be solved for latest-edge now
1885    It's everything. You simply have to delete from config this
lines which you have error. It's now done by zigbee2mqtt itself. I
reduce the config-Code to ''' data_path: /config/zigbee2mqtt socat:
e...
1955    I have the exact same problem after rebooting my system. Two
Tradfri E1810 (remote Ikea) stopped working. Reconnecting doesn't work
for either. But all other devices worked normally. When a lamp w...
2566    And this is still in the log: ''' 21-03-02 15:19:14 ERROR
(SyncWorker_3) Image zigbee2mqtt/zigbee2mqtt-aarch64 not exists for
addon_45df7312_zigbee2mqtt 21-03-02 15:23:58 INFO (SyncWorker_4)
Start...
Name: normalized_text, dtype: object

from sklearn.decomposition import NMF

nmf_text_model = NMF(n_components=10, random_state=42)
W_text_matrix = nmf_text_model.fit_transform(tfidf_text_vectors)
H_text_matrix = nmf_text_model.components_

```

El tema de un texto viene dado por la distribución de las palabras que contiene, por lo tanto, analizar esta distribución puede ser de gran ayuda para descubrir los temas.

Haciendo uso de la matriz H, se debe encontrar el índice de los mayores valores de cada fila para luego ser utilizado como índice de búsqueda en el vocabulario.

Se definirá una función que evalúe el modelo y muestre por pantalla un resumen de los temas (topics) que NMF ha detectado en los textos.

```
def display_topics(model, features, no_top_words=5):
    for topic, words in enumerate(model.components_):
        total = words.sum()
        largest = words.argsort()[:-1] # invert sort order
        print("\nTopic %02d" % topic)
        for i in range(0, no_top_words):
            print("  %s (%2.2f)" % (features[largest[i]],
abs(words[largest[i]]*100.0/total)))

display_topics(nmf_text_model,
tfidf_text_vectorizer.get_feature_names_out())
```

Topic 00

stale (18.06)
days (17.31)
activity (8.96)
label (8.96)
comment (8.86)

Topic 01

add (2.32)
z2m (2.01)
version (1.86)
ha (1.43)
new (1.23)

Topic 02

thanks (50.82)
worked (3.38)
working (1.87)
lot (1.09)
reply (0.99)

Topic 03

issue (41.26)
having (2.15)
got (1.62)
close (1.11)
exact (1.07)

Topic 04

url (29.25)
duplicate (4.55)
related (2.59)

```
closing (1.28)  
device (1.22)
```

Topic 05

```
error (3.03)  
zigbee (2.55)  
zigbee2mqtt (2.52)  
herdsman (2.11)  
info (2.04)
```

Topic 06

```
problem (21.86)  
solution (3.05)  
solved (2.78)  
zha (1.10)  
thank (1.09)
```

Topic 07

```
config (4.15)  
zigbee2mqtt (2.28)  
configuration (2.24)  
yaml (2.03)  
addon (1.85)
```

Topic 08

```
fixed (16.32)  
edge (5.29)  
dev (4.58)  
branch (4.31)  
latest (3.39)
```

Topic 09

```
update (13.43)  
42 (3.94)  
working (1.91)  
version (1.40)  
updated (1.36)
```

La salida muestra las palabras más relevantes para cada tema, permitiendo determinar la temática de algunos de los comentarios. Por ejemplo, en *Topic 02* se puede deducir que se trata de un mensaje de agradecimiento como respuesta a otro y en *Topic 04* que se trata de algún problema con una URL concreta.

Sería interesante conocer como de grande es una temática, es decir, cuantos comentarios están relacionados con un mismo tema por ejemplo.

Esto se puede calcular utilizando la matriz de temas de un documento y sumando las contribuciones individuales a este a lo largo de todos los documentos del data frame.

```
W_text_matrix.sum(axis=0)/W_text_matrix.sum()*100.0
```

```
array([18.15077823, 11.61180167, 5.36082748, 9.99092406,
8.43365604,
     10.84861004, 7.45875981, 12.96782206, 6.9400853 ,
8.23673532])
```

Este resultado indica que hay temas de mayor y menor peso pero no hay grandes diferencias en los porcentajes, indicando una supuesta buena calidad al tener una distribución bastante similar.

Creación de Modelos temáticos utilizando SVD

Otro algoritmo utilizado para el modelo de temas es el basado en la *descomposición de valores singular (SVD)*

Se trata de una técnica de factorización matricial utilizada para reducir la dimensionalidad de los datos y extraer patrones semánticos latentes. En este método, la matriz de datos es descompuesta en tres matrices: U , que representa la relación entre documentos y conceptos latentes; Σ , que contiene los valores singulares que indican la importancia de cada concepto; y V^T , que relaciona términos con estos conceptos.

En el procesamiento de lenguaje natural habitualmente se utiliza una variante de este algoritmo denominada *Truncated SVD* que reduce considerablemente el costo computacional.

```
from sklearn.decomposition import TruncatedSVD

svd_text_model = TruncatedSVD(n_components = 10, random_state=42)
W_svd_text_matrix = svd_text_model.fit_transform(tfidf_text_vectors)
H_svd_text_matrix = svd_text_model.components_
```

La función antes definida para evaluar y mostrar por pantalla el modelo puede ser reutilizado para este método.

```
display_topics(svd_text_model,
tfidf_text_vectorizer.get_feature_names_out())
```

```
Topic 00
stale (15.99)
days (15.34)
activity (7.94)
label (7.94)
comment (7.85)
```

```
Topic 01
zigbee2mqtt (1.53)
issue (1.20)
_url_ (1.04)
config (0.94)
error (0.88)
```

```
Topic 02
```

thanks (52.50)
issue (8.42)
url (6.00)
worked (3.65)
working (2.58)

Topic 03
issue (17.00)
url (13.62)
duplicate (2.07)
fixed (2.04)
related (1.32)

Topic 04
url (65.02)
duplicate (10.46)
stale (6.17)
days (5.29)
related (5.24)

Topic 05
url (8.70)
issue (7.85)
error (7.07)
herdsman (6.39)
zigbee (6.35)

Topic 06
problem (1525.35)
solution (208.44)
solved (189.74)
version (145.35)
zigbee (143.70)

Topic 07
update (50.30)
version (49.80)
edge (36.24)
latest (18.68)
fixed (15.63)

Topic 08
fixed (119.10)
dev (55.34)
edge (51.06)
branch (39.53)
z2m (32.51)

Topic 09
addon (6.22)

```

zigbee2mqtt (5.33)
fixed (4.58)
update (4.50)
config (3.81)

svd_text_model.singular_values_
array([18.83292308,  8.18662956,  6.33037565,  6.15879161,
5.60076025,
      5.34577855,  4.83188556,  4.51548091,  4.19867845,
4.0088322 ])

```

Al igual que con NMF, los valores singulares obtenidos representan la importancia relativa de cada componente latente en la estructura del texto. Un valor más alto indica que el componente correspondiente explica una mayor variabilidad en los datos, lo que sugiere que los temas asociados a esos componentes son más representativos en el corpus. A medida que los valores disminuyen, los temas capturados tienen menor influencia en la distribución global de términos y documentos. Esto permite reducir la dimensionalidad del texto, reteniendo solo las estructuras más significativas para el análisis.

Ambos métodos vistos utilizan métodos algebraicos usando como base la matriz de términos para la descomposición de temas. A partir de este punto, se verán modelos probabilísticos cuya popularidad ha aumentado en los últimos años.

Asignación latente de Dirichlet (LDA) con scikit-learn

LDA es el método de modelado de tópicos más popular en los últimos tiempos gracias a su capacidad de adaptación en diferentes escenarios.

Su funcionamiento consiste en leer cada documento como una mezcla de diferentes temas que a su vez son una mezcla de palabras. Para mantener una cantidad reducida de estos hace uso de la distribución Dirichlet que asegura que cada documento esté compuesto por unos pocos tópicos que al mismo tiempo están compuestos por una cantidad reducida de palabras.

Creación de Modelos temáticos utilizando LDA

Para este tipo de modelos también se utilizará la librería *scikit-learn*. Debido a que se hace uso de un método probabilístico, la duración del proceso es mayor en comparación con NMF y SVD.

```

from sklearn.feature_extraction.text import CountVectorizer
count_text_vectorizer = CountVectorizer(stop_words=list(stopwords),
min_df=5, max_df=0.7)
count_text_vectors =
count_text_vectorizer.fit_transform(df['normalized_text'])
count_text_vectors.shape
(2678, 1721)

```

```
from sklearn.decomposition import LatentDirichletAllocation

lda_text_model = LatentDirichletAllocation(n_components = 10,
random_state=42)
W_lda_text_matrix = lda_text_model.fit_transform(count_text_vectors)
H_lda_text_matrix = lda_text_model.components_
```

En este caso también se puede utilizar la función antes definida para la evaluación y muestra de los resultados:

```
display_topics(lda_text_model,
count_text_vectorizer.get_feature_names_out())
```

Topic 00

```
add (2.30)
version (1.94)
zigbee2mqtt (1.93)
addon (1.74)
issue (1.71)
```

Topic 01

```
false (4.80)
null (3.89)
failed (2.97)
zigbee2mqtt (2.81)
state (2.71)
```

Topic 02

```
ezsp (5.00)
zigbee (4.44)
01 (4.30)
adapter (3.80)
herdsman (3.63)
```

Topic 03

```
days (13.49)
stale (12.90)
issue (7.89)
open (7.75)
remove (7.53)
```

Topic 04

```
zigbee (4.76)
herdsman (4.03)
zigbee2mqtt (3.47)
error (3.40)
adapter (2.97)
```

Topic 05

```

config (3.35)
mqtt (2.77)
file (2.59)
port (2.54)
zigbee2mqtt (2.30)

Topic 06
info (5.37)
root (3.65)
zigbee2mqtt (3.59)
22 (2.16)
11 (1.39)

Topic 07
z2m (2.32)
_url_ (2.00)
problem (1.93)
issue (1.72)
device (1.36)

Topic 08
2022 (7.85)
08 (6.97)
10 (5.63)
zigbee2mqtt (5.55)
30 (5.29)

Topic 09
mainthread (7.16)
10 (6.33)
zigbee2mqtt (4.70)
option (4.37)
warning (4.32)

W_lda_text_matrix.sum(axis=0)/W_lda_text_matrix.sum()*100.0
array([32.30908695,  3.43165214,  3.47439399, 16.02876733,
5.04281378,
       6.09383153,  5.47115736, 22.13451431,  2.4828182 ,
3.53096439])

```

A simple vista se aprecia que LDA ha generado una estructura de temas bastante diferente en comparación con las anteriores. Además, en los resultados hay un porcentaje que destaca por ser muy grande en comparación con los demás, indicando así que la calidad del modelo puede mejorarse pues la distribución no es muy equivalente. Esto se podría conseguir variando el número de documentos elegidos para la ejecución hasta conseguir una distribución con menos diferencia entre los resultados que se obtengan.

Visualización de los resultados de LDA

La librería pyLDAvis es útil para la visualización de los resultados del modelado LDA tomando directamente los resultados obtenidos previamente.

```
import pyLDAvis
import pyLDAvis.lda_model

lda_display = pyLDAvis.lda_model.prepare(
    lda_text_model, # Modelo LDA entrenado
    count_text_vectors, # Matriz documento-término
    count_text_vectorizer, # Vectorizador (para obtener vocabulario)
    sort_topics=False
)

pyLDAvis.display(lda_display)
<IPython.core.display.HTML object>

lda_tsne_display = pyLDAvis.lda_model.prepare(lda_text_model,
count_text_vectors, count_text_vectorizer, sort_topics=False,
mds='tsne')
pyLDAvis.display(lda_tsne_display)
<IPython.core.display.HTML object>
```

La visualización generada por pyLDAvis ofrece una representación gráfica de los temas extraídos de un conjunto de documentos. En ella, cada tema se representa como un círculo cuyo tamaño indica su prominencia en el conjunto de datos, mientras que la posición refleja su relación con otros temas. Las palabras más frecuentes y representativas de cada tema se muestran junto al círculo correspondiente, lo que permite interpretar el contenido de los temas. Esta visualización facilita la comprensión de las principales categorías presentes en los textos, ayuda a identificar patrones y relaciones entre los temas, y es útil para asignar documentos a sus temas más relevantes o explorar soluciones a problemas comunes en grandes volúmenes de texto.

Utilización de nubes de palabras para mostrar y comparar modelos

En capítulos anteriores ya se han visto y explicado lo útiles que pueden ser las nubes de palabras para comparar las diferencias entre los resultados de unos y otros modelos.

Se generarán ahora estas nubes para ver las que hay entre los modelos NMF y LDA.

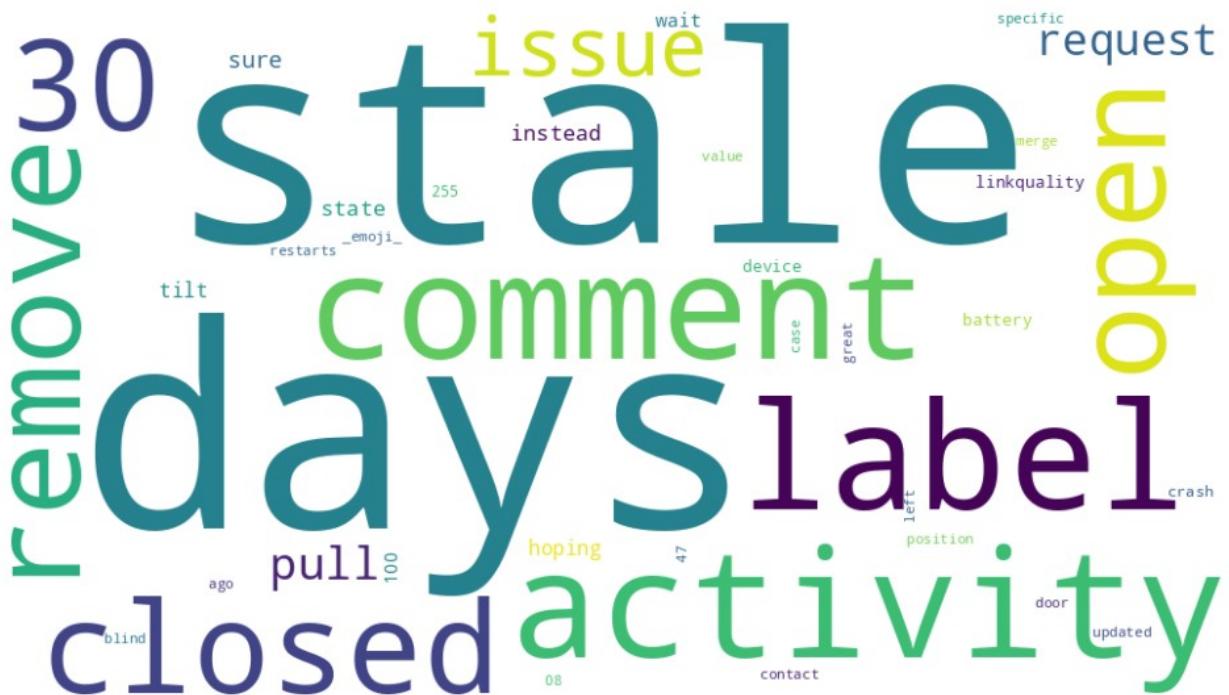
```
#%matplotlib inline
import matplotlib.pyplot as plt
from wordcloud import WordCloud

def wordcloud_topics(model, features, no_top_words=40):
    for topic, words in enumerate(model.components_):
        size = {}
        largest = words.argsort()[-1] # invert sort order
```

```
for i in range(0, no_top_words):
    size[features[largest[i]]] = abs(words[largest[i]])
wc = WordCloud(background_color="white", max_words=100,
width=960, height=540)
wc.generate_from_frequencies(size)
plt.figure(figsize=(12,12))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
# if you don't want to save the topic model, comment the next
line
# plt.savefig(f'topic{topic}.png')

wordcloud_topics(nmf_text_model,
tfidf_text_vectorizer.get_feature_names_out())

/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
matplotlib_inline/config.py:68: DeprecationWarning:
InlineBackend._figure_format_changed is deprecated in traitlets 4.1:
use @observe and @unobserve instead.
def _figure_format_changed(self, name, old, new):
```



repository installed running
like doesn't supervisor core old work
need device
issues add new devices home
zigbee2mqtt supervisor edge addon repo sure
fix works backup restart os try latest
assistant ha edge Z2m image working

image looks tomorrow
support sorry testing change
yes docs reply nice
matter lot hi newbie
forgot view v1 help
sense end reporting didn't
work docker access spot project
like working worked reporting
updated report
close good koenkk know
stupid understand

issue 40

exact exists having coordinator core figure looks news addons addon report latest address think solution similar relevant experienced seen zigbee2mqtt working tracker close got past relevant sonoff fix know ha solved changed like reported sonoff coordinator experiencing seen zigbee2mqtt

firmware open repository add provide look log specific create duplicate try logging enable frontend repo feature support exists maybe continue like

report device url - issues help related note use probably sure let log probably create use specific note feature support exists maybe continue like

bug caused think reported help issues

zigbee2mqtt error or info
node_modules false 03 znp zstack 11
false 03 failed 2024 05
2023 07
2022 08
log 02
adapter 17
ezsp 2021 11
ts 07
controllerlib 18
controllerlib 01
src 12
herdsman 04
startstarting 14
exact unfortunatelly identical
found solve stick
thank dongle got zha updating
skyconnect
hello solved ve
close 37 use don ok hi
integration 37 use read able
databasetry find browser
problem broker fixes
cache reboot sonoff solution help
reboot sonoff solution help
similar

homeassistant true false uioptions server
zigbee2mqtt copy tcp
folder serial adapter devices
configuration tab groups
file frontend
dev mqtt addon
default user link old
user change password files
addon_configs share
config yaml socat json data_path
mqtt port

adapter version configuration
closing adding _emoji_
koenkk stable
added updating
added fix work
closes report
fixes commit
fixes 17
fixes yes
fixes hours
fixes check
fixes changes
fixes versions
fixes available
fixes cache
fixes mins
fixes pushed
fixes ezsp
fixes master
fixes chrome
fixes release
fixes true
fixes error
fixes branch
fixes soon
fixes serial
fixes mean
fixes pr
fixes latest
fixes devedge



```
wordcloud_topics(lda_text_model,  
count_text_vectorizer.get_feature_names_out())
```



A word cloud visualization centered around Zigbee2MQTT configuration terms. The most prominent words are "topic" (large blue), "payload" (green), "failed" (purple), "state" (blue), "device" (green), and "router" (yellow). Other visible words include "publish" (purple), "subscribe" (blue), "data" (dark blue), "error" (green), "nuisance" (blue), "hue" (blue), "color" (green), "type" (green), "52" (blue), "direction" (blue), "response" (green), "disabled" (blue), "brightness" (yellow), "srcendpoint" (blue), "ieeeaddr" (blue), "addon_configs" (blue), "transactionsequence" (blue), "set" (blue), "timeout" (green), "04" (green), "10000" (blue), "single" (blue), "model" (blue), "writeundiv" (blue), "true" (blue), "command" (green), "topic" (blue), "manufacturercode" (blue), "genonoff" (blue), "2023" (yellow), and "linkquality" (green).

closed issue stale reverse
homekit nodon fork containers approach pretty happen
nginx exposed followed pull left provide tilt date
google months support brand built
position copying 30 expose spot remove
comment called readme saw
label

11 app devices node_modules
herdsman npm
07 ts error converters
dev log lib znp 02 adapter
serialport const err unpi sys failed
05 29 00 zigbee 03 start
controller 2022 zigbee2mqtt
js 12 zstack src

A word cloud visualization centered around the word "config". The words are colored in various shades of green, blue, and yellow, and are surrounded by small text labels indicating their context. The words include "homeassistant", "mqtt", "yaml", "serial", "options", "usr", "dev", "channel", "true", "ui", "zigbee2mqtt", "core", "friendly_name", "mode", "keepalive", "password", "user", "configuration", "data_path", "false", "socat", "server", "groups", "base_topic", "port", "line", "local", "frontend", "py", "log", "11", "tcp", "advanced", "master", "devices", and "link".

release configuration settings
logs work fixed
has switch looks value
addon coordinator works
restart coordinator
think working
need device
tried ve worked set
sensor button try thanks
yaml 11 problem time
mqtt frontend
tried solution issues don

issue

z2m

problem

topic zigbee2mqtt
payload
2022
1008
info
attributereport
zigbee
state
message
groupid 05
publish
endpoint
debug cluster open
battery
last_seen
elapsed
closureswindowcovering
received
position
data
linkquality



Modelado de temas utilizando Gensim

Otra herramienta que se puede utilizar para la creación de modelos temáticos es Gensim, ofrece una mayor variedad de algoritmos para los cálculos de modelos en comparación con *scikit-learn* y también puede proporcionar estimaciones sobre la calidad de los mismos.

Preparación de los datos

En primer lugar se debe preparar el vocabulario. Se debe realizar una tokenización manual porque Gensim no cuenta con uno integrado y la herramienta espera que todas las líneas estén ya tokenizadas.

En este caso, ya están todos los documentos tokenizados como parte de los primeros pasos que se tomaron en la realización del proyecto. Como muestra, se pueden imprimir 5 documentos aleatorios tokenizados de la columna *tokens* del Data Frame:

```
print(df['tokens'].sample(5))

2555
I'm,using,zzh,and,it's,been,working,now,with,latest,updates,1.18.1-1
767
Hi,ghork125,Is,your,solution,actual,for,Aqara,D1,N0,NEUTRAL,version,only,I,have,the,same,problem,with,NEUTRAL,version,May,I,use,the,same,drivers, and,the,same,steps,to,try,to,fix,this,prblem,Thank,....
26
I'm,totally,lost,Have,been,trying,to,restore,everything,but,the,Addon,won't,start,due,to,502,Bad,Gateway,:-
(,INFO,Preparing,to,start,INFO,Socat,not,enabled,INFO,Starting,Zigbee2
```

```
MQTT,Starting,Zigbe...
1087
I,had,similar,problem,Total,reboot,helped
553           But,it,looks,like,the,app,node_modules,winston-
transport,node_modules,readable-
stream,lib,_stream_writable.js:264,var,er,new,ERR_STREAM_WRITE_AFTER_E
ND,is,gone,now,No,I've,updated,the,post,above
Name: tokens, dtype: object
```

Debido a que en la columna *tokens* del data frame se almacenan los mismos en formato de cadena y gensim espera que estén en formato lista, se van a volver a tokenizar los textos para obtener el diccionario.

```
#tokenización
gensim_tokens = [[w for w in re.findall(r'\b\w\w+\b' ,
comment.lower()) if w not in stopwords]
                  for comment in df["text"]]

from gensim.corpora import Dictionary

dict_gensim_comments = Dictionary(gensim_tokens)
```

Es recomendable reducir las dimensiones del vocabulario filtrando las palabras que aparecen con demasiada o muy poca frecuencia. Se definirá un mínimo de 5 documentos en los que las palabras deben aparecer pero que no lo hagan en más de un 70% para mantener unas dimensiones contenidas.

```
dict_gensim_comments.filter_extremes(no_below=5, no_above=0.7)
```

Se realiza el cálculo de la matriz de bag-of-words.

```
bow_gensim_comments = [dict_gensim_comments.doc2bow(comment) for
comment in gensim_tokens]
```

Para finalmente realizar la transformación TF-IDF.

```
from gensim.models import TfidfModel

tfidf_gensim_comments = TfidfModel(bow_gensim_comments)
vectors_gensim_comments = tfidf_gensim_comments[bow_gensim_comments]
```

La matriz vectors_gensim_comments es la que se usará para las siguientes tareas.

Factorización de la Matriz No Negativa (NMF)

Se calculará la Matriz No Negativa de forma similar a cómo se hacía con scikit-learn y se observarán los resultados obtenidos para comprobar si son similares a los obtenidos con anterioridad.

```

from scipy.sparse import csc_matrix
from gensim.models.nmf import Nmf

nmf_gensim_comments = Nmf(vectors_gensim_comments, num_topics=10,
id2word=dict_gensim_comments, kappa=0.1, eval_every=5,
random_state=42)

/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:578: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`-
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:607: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`-
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:620: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`-
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:620: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`-
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):

```

Se define una función que muestre por pantalla los resultados obtenidos:

```

def display_topics_gensim(model):
    for topic in range(0, model.num_topics):
        print("\nTopic %02d" % topic)
        for (word, prob) in model.show_topic(topic, topn=5):
            print("  %s (%.2f)" % (word, prob))

display_topics_gensim(nmf_gensim_comments)

```

```

Topic 00
issue (0.05)
config (0.02)
thanks (0.02)
fixed (0.02)
2023 (0.01)

Topic 01
stale (0.12)
days (0.11)
activity (0.06)

```

```
label (0.06)
comment (0.06)
```

```
Topic 02
thanks (0.05)
koenkk (0.03)
com (0.03)
issues (0.03)
https (0.03)
```

```
Topic 03
edge (0.02)
error (0.01)
info (0.01)
device (0.01)
2022 (0.01)
```

```
Topic 04
zigbee2mqtt (0.04)
github (0.04)
issues (0.03)
com (0.03)
https (0.03)
```

```
Topic 05
stale (0.13)
days (0.12)
activity (0.06)
label (0.06)
comment (0.06)
```

```
Topic 06
z2m (0.02)
add (0.02)
devices (0.02)
mqtt (0.02)
try (0.01)
```

```
Topic 07
configuration (0.02)
yaml (0.02)
adapter (0.01)
user (0.01)
debug (0.01)
```

```
Topic 08
zigbee2mqtt (0.03)
addon (0.03)
hassio (0.02)
issuecomment (0.01)
```

```
system (0.01)

Topic 09
assistant (0.02)
home (0.02)
version (0.02)
usb (0.02)
2024 (0.01)
```

Se observa que los resultados no son idénticos, pero si cuentan con algunas similitudes. Pese a que los porcentajes obtenidos son menores, las palabras asociadas a los temas sí tienen sentido teniendo en cuenta que para el entrenamiento del modelo no se han utilizado necesariamente los mismos comentarios que los utilizados con anterioridad.

Gensim también cuenta con funciones para calcular la puntuación de coherencia de los *topics*, que puede ser un medidor de calidad del modelo.

```
from gensim.models.coherencemodel import CoherenceModel

nmf_gensim_comms_coherence = CoherenceModel(model=nmf_gensim_comments,
texts=gensim_tokens, dictionary=dict_gensim_comments, coherence='c_v')
nmf_gensim_comms_coherence_score =
nmf_gensim_comms_coherence.get_coherence()

print(nmf_gensim_comms_coherence_score)

0.4488647325778645
```

Se obtiene una puntuación de casi el 45%, lo que indica un modelo de no muy buena calidad.

Siempre se pueden variar los valores con los que se entrena el modelo para lograr un mejor rendimiento, algo que es altamente recomendable y se hará a continuación.

```
#Reducción de dimensiones del vocabulario
dict_gensim_comments.filter_extremes(no_below=3, no_above=0.7)

#Cálculo de la matriz bag-of-words
bow_gensim_comments = [dict_gensim_comments.doc2bow(comment) for
comment in gensim_tokens]

#Transformación TF-IDF
tfidf_gensim_comments = TfidfModel(bow_gensim_comments)
vectors_gensim_comments = tfidf_gensim_comments[bow_gensim_comments]

#NMF
nmf_gensim_comments = Nmf(vectors_gensim_comments, num_topics=10,
id2word=dict_gensim_comments, kappa=0.5, eval_every=10,
random_state=42)
```

```
#Resultados
display_topics_gensim(nmf_gensim_comments)

/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:578: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:607: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:620: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
/home/diego/.pyenv/versions/gvtiaenv/lib/python3.10/site-packages/
gensim/models/nmf.py:620: DeprecationWarning: Please import
`csc_matrix` from the `scipy.sparse` namespace; the `scipy.sparse.csc`
namespace is deprecated and will be removed in SciPy 2.0.0.
    if isinstance(corpus, scipy.sparse.csc.csc_matrix):
```

Topic 00

```
issue (0.04)
config (0.02)
fixed (0.01)
thanks (0.01)
2023 (0.01)
```

Topic 01

```
stale (0.09)
days (0.09)
activity (0.05)
label (0.05)
comment (0.05)
```

Topic 02

```
thanks (0.08)
koenkk (0.03)
issues (0.03)
com (0.02)
https (0.02)
```

Topic 03

```
edge (0.02)
info (0.01)
error (0.01)
```

```
device (0.01)
2022 (0.01)

Topic 04
issues (0.04)
github (0.04)
zigbee2mqtt (0.03)
com (0.03)
koenkk (0.03)

Topic 05
stale (0.10)
days (0.09)
activity (0.05)
label (0.05)
comment (0.05)

Topic 06
devices (0.01)
add (0.01)
z2m (0.01)
mqtt (0.01)
yaml (0.01)

Topic 07
configuration (0.01)
yaml (0.01)
debug (0.01)
user (0.01)
adapter (0.01)

Topic 08
zigbee2mqtt (0.03)
addon (0.03)
hassio (0.03)
github (0.02)
com (0.02)

Topic 09
assistant (0.02)
home (0.02)
usb (0.01)
version (0.01)
2024 (0.01)

#Coherencia y calidad del modelo
nmf_gensim_comms_coherence = CoherenceModel(model=nmf_gensim_comments,
texts=gensim_tokens, dictionary=dict_gensim_comments, coherence='c_v')
nmf_gensim_comms_coherence_score =
nmf_gensim_comms_coherence.get_coherence()
```

```
print(nmf_gensim_comms_coherence_score)  
0.45328958792997165
```

Tras realizar pruebas con distintos valores, se ha conseguido elevar la puntuación levemente y se aprecian pequeñas diferencias respecto a las palabras que componen cada uno de los temas.

LDA

El modelado *LDA* junto con *Gensim* es muy sencillo de realizar, sobre todo si los datos ya están preparados como es el caso.

Se utiliza la clase *LdaModel* junto con una gran variedad de parámetros para la creación del modelo. Por ahora se utilizarán los valores recomendados para los parámetros.

```
from gensim.models import LdaModel  
  
lda_gensim_comms = LdaModel(corpus=bow_gensim_comments,  
id2word=dict_gensim_comments, chunksize=2000,  
    alpha='auto', eta='auto', iterations=400, num_topics=10,  
passes=20, eval_every=None, random_state=42)  
  
display_topics_gensim(lda_gensim_comms)  
  
Topic 00  
issue (0.03)  
problem (0.02)  
add (0.02)  
version (0.02)  
ha (0.02)  
  
Topic 01  
zigbee2mqtt (0.05)  
error (0.05)  
app (0.04)  
js (0.03)  
zigbee (0.03)  
  
Topic 02  
days (0.13)  
stale (0.11)  
open (0.08)  
issue (0.08)  
closed (0.07)  
  
Topic 03  
2021 (0.06)  
zigbee (0.06)
```

```
adapter (0.06)
herdsman (0.06)
zstack (0.02)
```

Topic 04

```
type (0.03)
true (0.03)
mqtt (0.02)
zigbee2mqtt (0.02)
description (0.02)
```

Topic 05

```
zigbee2mqtt (0.03)
config (0.03)
new (0.02)
addon (0.02)
add (0.02)
```

Topic 06

```
false (0.04)
error (0.03)
null (0.03)
failed (0.03)
2024 (0.02)
```

Topic 07

```
dev (0.11)
00 (0.05)
devices (0.05)
02 (0.05)
usb (0.03)
```

Topic 08

```
2022 (0.07)
10 (0.06)
zigbee2mqtt (0.06)
08 (0.06)
info (0.06)
```

Topic 09

```
zigbee2mqtt (0.06)
https (0.04)
com (0.03)
config (0.03)
github (0.03)
```

Resulta más complicado de interpretar los temas obtenidos en esta ocasión en comparación con los calculados por NMF.

Gensim permite calcular el valor de la complejidad, útil para medir cómo de bien un modelo predice una muestra.

```
lda_gensim_comms.log_perplexity(vectors_gensim_comments)  
-8.727633594706381
```

Puntuación de coherencia

Al igual que con el método NMF, también es posible calcular la puntuación de coherencia de un modelo que hace uso de LDA.

La librería ofrece una herramienta que encapsula todos los pasos necesarios para el cálculo:

```
from gensim.models.coherencemodel import CoherenceModel  
  
lda_gensim_comms_coherence = CoherenceModel(model=lda_gensim_comms,  
texts=gensim_tokens, dictionary=dict_gensim_comments, coherence='c_v')  
lda_gensim_comms_coherence_score =  
lda_gensim_comms_coherence.get_coherence()  
  
print(lda_gensim_comms_coherence_score)  
0.5459668633868128
```

Simplemente sustituyendo *lda* por *nmf* se puede calcular la coherencia del modelo NMF y así comparar los resultados.

```
from gensim.models.coherencemodel import CoherenceModel  
  
nmf_gensim_comms_coherence = CoherenceModel(model=nmf_gensim_comments,  
texts=gensim_tokens, dictionary=dict_gensim_comments, coherence='c_v')  
nmf_gensim_comms_coherence_score =  
nmf_gensim_comms_coherence.get_coherence()  
  
print(nmf_gensim_comms_coherence_score)  
0.45328958792997165
```

El resultado obtenido con NMF es similar al del modelo LDA.

Calcular la coherencia de los temas individualmente con el modelo LDA es aún más sencillo porque esta función se encuentra ya implementada en el propio modelo.

```
top_topics = lda_gensim_comms.top_topics(vectors_gensim_comments,  
topn=5)  
avg_topic_coherence = sum([t[1] for t in top_topics]) /  
len(top_topics)  
  
print('Average topic coherence: %.4f.' % avg_topic_coherence)
```

Average topic coherence: -1.3260.

Es posible mostrar la coherencia de los temas individualmente junto con las palabras más importantes de cada uno:

```
[[(t[1], " ".join([w[1] for w in t[0]])) for t in top_topics]
[(-0.1784010715957743, 'days stale open issue closed'),
 (-1.0022238132246468, 'zigbee2mqtt https com config github'),
 (-1.134095121441248, '2022 10 zigbee2mqtt 08 info'),
 (-1.1394033211397536, 'false error null failed 2024'),
 (-1.1865445168569964, 'zigbee2mqtt error app js zigbee'),
 (-1.4253368921017446, '2021 zigbee adapter herdsman zstack'),
 (-1.4944864843749825, 'zigbee2mqtt config new addon add'),
 (-1.6489372253081391, 'type true mqtt zigbee2mqtt description'),
 (-1.941653173417604, 'issue problem add version ha'),
 (-2.1086963893838315, 'dev 00 devices 02 usb')]
```

Búsqueda del número óptimo de temas

Es difícil conocer el número correcto de temas que se deben utilizar para un modelo pues supondría una gran cantidad de tiempo variar los valores de los parámetros mediante prueba y error, guardar estos valores junto con los resultados obtenidos y luego comparar manualmente cuál ofrece un mejor desempeño.

Para esta tarea, Gensim ofrece la posibilidad de calcular este valor mediante la clase *LdaMulticore*.

Se debe tener en cuenta que este proceso necesita de un tiempo de ejecución considerablemente mayor en comparación a los otros vistos previamente porque crea tantos modelos como el usuario le indica en el parámetro *range* en el bucle for. En este caso, se han creado 16 modelos (rango 5-21) que ha conllevado un tiempo de ejecución de 7 minutos aproximadamente.

```
from gensim.models.ldamulticore import LdaMulticore

lda_comms_model_n = []
for n in tqdm(range(5, 21)):
    lda_model = LdaMulticore(corpus=bow_gensim_comments,
id2word=dict_gensim_comments, chunksize=2000,
                           eta='auto', iterations=400, num_topics=n,
passes=20,
                           eval_every=None, random_state=42)
    lda_coherence = CoherenceModel(model=lda_model,
texts=gensim_tokens,
                           dictionary=dict_gensim_comments,
coherence='c_v')
    lda_comms_model_n.append((n, lda_model,
lda_coherence.get_coherence()))
```

Es posible generar un **gráfico de coherencia** para evaluar la calidad del modelo LDA en función del número de temas utilizado. Esto permite determinar el número óptimo de temas para LDA, buscando un punto en el que la coherencia sea máxima antes de estabilizarse o incluso reducirse.

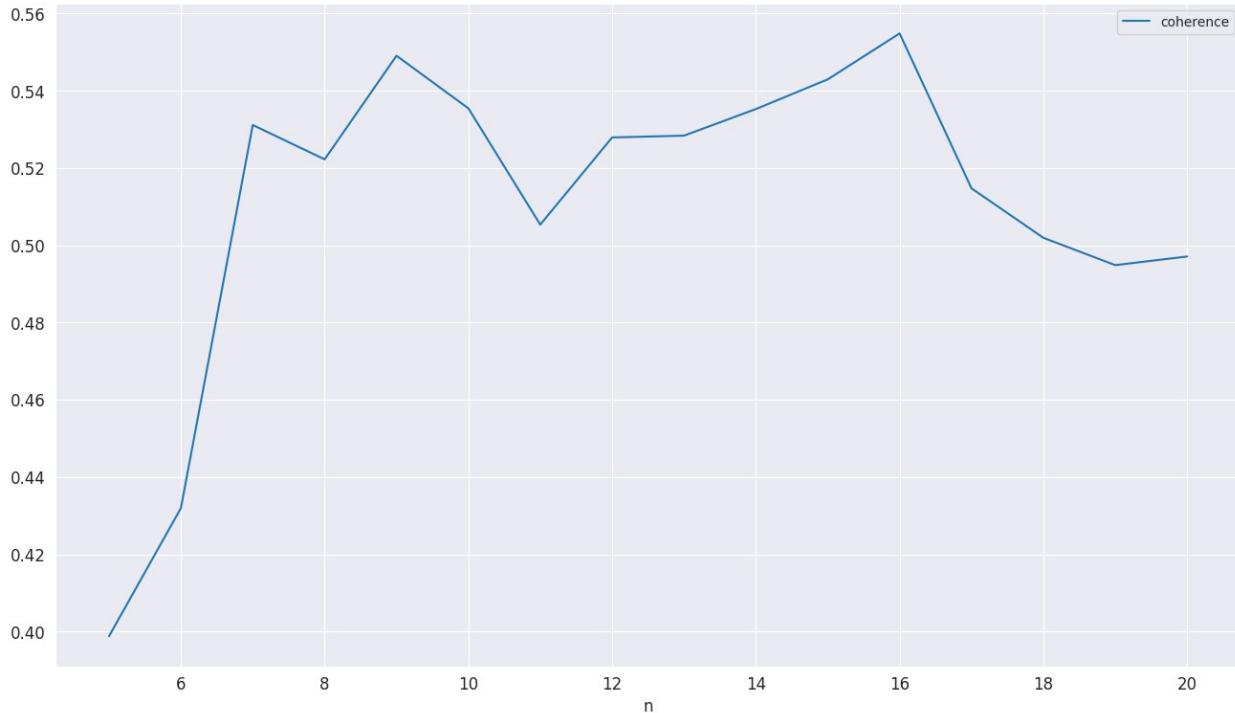
En el proceso se crea un DataFrame con los valores almacenados en *lda_comms_model_n* que contiene:

- **n**: número de temas utilizados por el modelo
- **model**: modelo LDA correspondiente a cada número de temas
- **coherence**: valor de coherencia del modelo - indicador de calidad

Para el DataFrame se establece *n* como índice para facilitar la visualización de cómo varía la coherencia en función de los temas utilizados.

```
pd.DataFrame(lda_comms_model_n, columns=["n", "model",
                                         "coherence"]).set_index("n")["coherence"].plot(figsize=(16,9))

<Axes: xlabel='n'>
```



A raíz del gráfico generado se observa con claridad que la mayor puntuación de coherencia se ha conseguido utilizando 17 temas obteniendo aproximadamente un 57%, más de un 10% con respecto al obtenido con NMF.

A continuación, se pueden mostrar los resultados de las palabras con mayor peso para cada tema para una comparación con los resultados obtenidos con el resto de modelos:

```
display_topics_gensim(lda_comms_model_n[12][1])
```

Topic 00

version (0.03)
issue (0.03)
update (0.02)
addon (0.02)
z2m (0.02)

Topic 01

user (0.06)
images (0.06)
com (0.05)
https (0.05)
githubusercontent (0.05)

Topic 02

days (0.16)
stale (0.15)
open (0.09)
issue (0.09)
closed (0.08)

Topic 03

config (0.03)
file (0.03)
configuration (0.02)
yaml (0.02)
true (0.02)

Topic 04

false (0.04)
null (0.04)
state (0.03)
failed (0.02)
2023 (0.02)

Topic 05

add (0.04)
zigbee2mqtt (0.04)
config (0.04)
addon (0.03)
new (0.02)

Topic 06

supervisor (0.05)
22 (0.04)
zigbee2mqtt (0.04)
mainthread (0.03)

10 (0.03)

Topic 07

port (0.03)
config (0.03)
ui (0.03)
serial (0.02)
dev (0.02)

Topic 08

2022 (0.08)
08 (0.07)
10 (0.06)
30 (0.06)
zigbee2mqtt (0.06)

Topic 09

zigbee2mqtt (0.08)
https (0.06)
github (0.05)
com (0.05)
koenkk (0.03)

Topic 10

zigbee2mqtt (0.04)
issue (0.02)
config (0.01)
directory (0.01)
homeassistant (0.01)

Topic 11

root (0.06)
info (0.05)
2022 (0.03)
08 (0.03)
zigbee (0.02)

Topic 12

ezsp (0.05)
01 (0.04)
adapter (0.04)
zigbee (0.03)
herdsman (0.03)

Topic 13

zigbee (0.05)
herdsman (0.05)
2023 (0.03)
ezsp (0.03)
const (0.03)

```
Topic 14
zigbee2mqtt (0.04)
zigbee (0.03)
error (0.03)
herdsman (0.03)
adapter (0.02)
```

```
Topic 15
issue (0.02)
system (0.02)
switch (0.02)
devices (0.02)
try (0.01)
```

```
Topic 16
problem (0.02)
2021 (0.02)
zigbee2mqtt (0.02)
thanks (0.02)
zigbee (0.02)
```

También se pueden generar las nubes de palabras para los temas obtenidos con este modelo:

```
def wordcloud_topics_gensim(model, no_top_words=40):
    for topic in range(0, model.num_topics):
        size = {}
        for (word, prob) in model.show_topic(topic,
topn=no_top_words):
            size[word] = prob
        wc = WordCloud(background_color="white", max_words=100,
width=960, height=540)
        wc.generate_from_frequencies(size)
        plt.figure(figsize=(12,12))
        plt.imshow(wc, interpolation='bilinear')
        plt.axis("off")
        # if you don't want to save the topic model, comment the next
line
        #plt.savefig(f'topic{topic}.png')

wordcloud_topics_gensim(lda_comms_model_n[12][1])
```

ciotlosm time running don looks fixed
edge ha error home works updated
z2m use fine
stable add release
version
working solution network updates latest
issues like problem new assistant os work
update addon firmware

11eb alt know change start github user content
error 22 control image lamp 03
configuration img 34 showing z2m width
config device 2022 debug
time
zigbee2mqtt
devices src 10 attachments remote
service 04 https
png yaml mqtt sensors

fork
remove
label
great
pull
disappeared
little
joshuaspence
30
edge
warning
is
final
wait
completed
zbdonglee
request
explanation
bar
rpi3b
contains
mins
blind
properly
activity
search
chromecast
closed
comment
safari
closed
issue
merged
merge
activity
search
chromecast
closed
comment
safari
closed

z2m
doesnt
yaml
true
don
json
configuration
state
key
problem
assistant
zigbee2mqtt
file
version
addon
use
options
type
sure
set
error
false
running
description
attributes
mqtt
default
availability
change
port
number
tilt
zha
home
ui
value
frontend
ha
ui
value
frontend
config

direction
device
disabledefaultresponse
error
topic
last_seen
fa
linkquality
52
command
line
statefailed
copy
repository
addon
work
home
share
config
42
case
zigbee2mqtt
use
gui
assistant
color_temp
null
publish
disableresponse
type
color
brightness
11
false
disableresponse
payload
file
timeout
10000
zigbee2mqtt
color_mode
model
homeassistant
mqtt
zclversion
15 py
configuration
version
update
dd
new
configure
container
install
remove
yaml
z2m
bind
old
devices
need
backup
repo
start
files
upgrade
addon_configs
file
assistant

option 18 edge 22 02 .15 30 error 33
connect default version 45df7312_zigbee2mqtt 2024
supervisor
12 13 23 05 warning 10 17 info
ingress exist starting schema
zigbee2mqtt docker type addons api 16 true
add **mainthread** options

version zigbee2mqtt fixed
edge true id add built
options configuration friendly name cache need assistant use instance
dev required lounge match devices usb settings
adding edit false frontend problem issues
host supervisor serial adapter yaml



A word cloud visualization showing the most common terms used in the Zigbee2MQTT GitHub repository. The words are colored in various shades of green, blue, and yellow, and are arranged in a dense cluster.

The most prominent words include:

- zigbee2mqtt
- github
- issues
- https
- com
- frontend
- www
- html
- configuration
- fixed
- core
- master
- guide
- fix
- devices
- dev
- pull
- log
- addon
- issuecomment
- io
- advanced
- addon
- device
- z2m
- assets
- changes
- debug
- logging
- image
- branch
- files
- information
- provide
- think
- issue
- think
- think

working like homeassistant
data_path addon_config
idea addon_configs
data addon found
work sorry changed
directory works config
problem wrong update
hassio running fine
ha different mqt
new update devices
time present thank
zigbee2mqtt
updated removed let
guess solved sensor

08 app 254 2022 12 adapter
herdsman init 06 11 dec
zigbee root 25
service 43 ok 32 zstack zigbee2mqtt.
starting 19 rc 14 09 s6 2024 js
10 19 lib znp node_modules
loader yaml unpi 16

core
port
raw
36
false
uart
zigbee2mqtt

55
5m
ttx2m
error
srclog
tcp
yaml

23
300
send
echo
data
31
303
19
14
13
config
user
server
failed

serial
queue

zigbeeherdsman

ezsp

012024 adapter

config
false
module
page
fz
18
43
exposes
database

tuya
extend
reporting
endpoint

herdsman

uart
Zigbee
converters

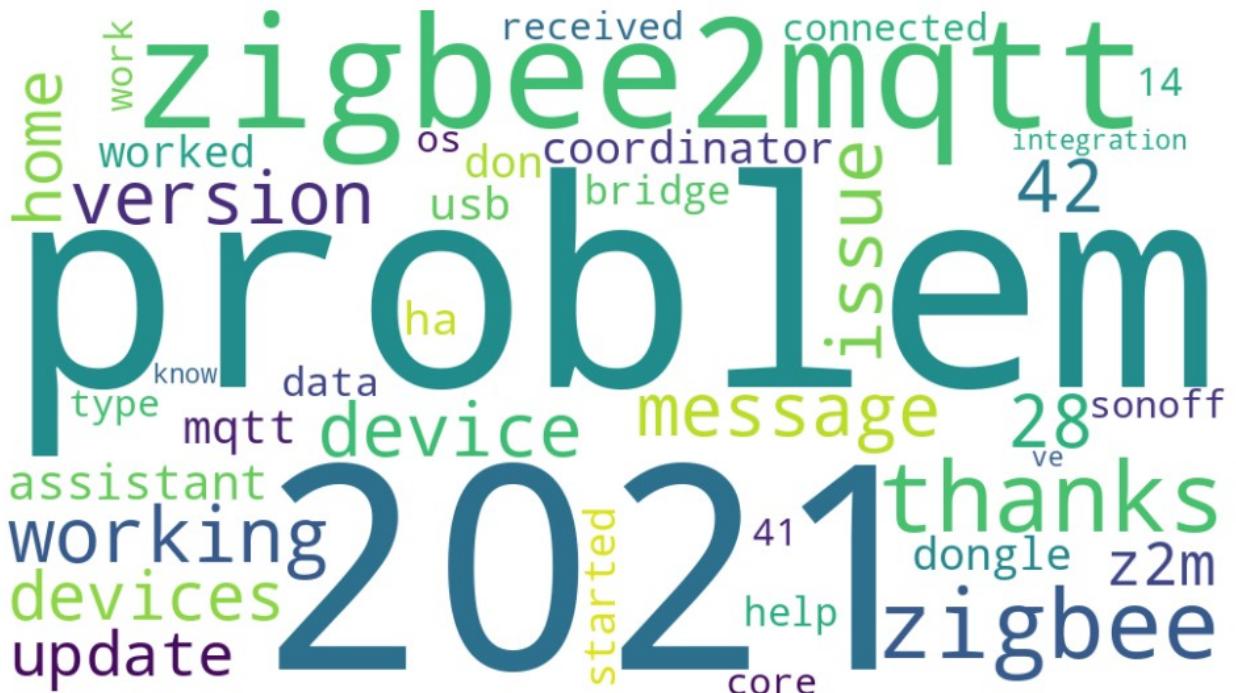
const
state
meta
ea
vendor
57
26
lib
model
56
skip
definition
12
value
+N
02

zigbeeherdsman

ezsp2023

adapter

node_modules 04 2024
adapter 01 17
start null 07 10
05 app znp 2021 03
18 controller failed 2023 10
herdsman error 12 10
14 ts config 00 03
zigbee2mqtt 221 02 11 03
zstack 29 03
starting 29 03



Creación de un Proceso Dirichlet Jerárquico (HDP)

HDP como extensión del método LDA permite visualizar los resultados en un formato consistente en dar unos pocos temas más amplios y bien diferenciados para luego proporcionar más detalles de estos incluyendo más palabras y subtemas.

Se trata de un método bastante nuevo, pero Gensim cuenta con una implementación experimental integrada que además permite hacer uso de la vectorización realizada antes.

```
from gensim.models import HdpModel  
  
hdp_gensim_comms = HdpModel(corpus=bow_gensim_comments,  
id2word=dict_gensim_comments)
```

HDP tiene la capacidad de hacer una estimación sobre el número de temas y luego mostrar todos aquellos que han sido identificados con la función `print_topics` y una serie de parámetros. Para mejorar la comprensión de los resultados, se puede crear un DataFrame con estos permitiendo una mejor visualización:

```

words = 8
pd.DataFrame([re.split(r" \+ |\\*", t[1]) for t in
hdp_gensim_comms.print_topics(num_topics=20, num_words=words)])

```

	0	1	2	3	4	\
0	0.029	zigbee2mqtt	0.017		2022	0.017
1	0.018	zigbee2mqtt	0.009		zigbee	0.007
2	0.010	color_temp	0.009		52	0.009
3	0.011	zigbee2mqtt	0.007		info	0.005

4	0.013	info	0.011	root	0.010		
5	0.020	root	0.005	data	0.005		
6	0.016	10	0.010	zigbee2mqtt	0.009		
7	0.007	2024	0.006	zigbee2mqtt	0.005		
8	0.005	zigbee2mqtt	0.004	02	0.004		
9	0.016	days	0.014	stale	0.009		
10	0.004	state	0.004	cc2652rb	0.004		
11	0.005	deleting	0.005	folders	0.004		
12	0.004	reinstalling	0.004	exposes	0.004		
13	0.005	110	0.004	error	0.004		
14	0.004	asking	0.004	press	0.004		
15	0.004	maintrel	0.003	aarch64	0.003		
16	0.005	bin	0.004	cc2652rb	0.004		
17	0.003	homeassistant	0.003	buttons	0.003		
18	0.005	com	0.004	rename	0.004		
19	0.004	automatically	0.004	update_available	0.003		
5 6 7 8 9							
10	\		5	6	7	8	9
0		zigbee	0.017	info	0.014		08
0.014		error	0.007	herdsman	0.006		https
0.006		04	0.009	2023	0.007	color_mode	
0.007		error	0.005	failed	0.005		false
0.005		ok	0.005	07	0.004	assistant	
0.004		blog	0.004	11	0.004		205
0.003		45df7312_zigbee2mqtt	0.009	option	0.009	supervisor	
0.008		info	0.005	ezsp	0.004		08
0.004		39	0.004	syntax	0.004	reinstalled	
0.003		30	0.008	closed	0.008		open
0.008		zigbee2mqtt	0.004	loaded	0.003	zstackadapter	
0.003		optional	0.004	exists	0.004		responds
0.003		suggest	0.003	switched	0.003		enum
0.003		zigbee2mqtt	0.004	2024	0.004		8888
0.003		guess	0.004	debug	0.003		cause
0.003							

15		34	0.003	exactly	0.003	humidity
0.003						
16	legacy_availability_payload	0.004	protocol	0.004		saved
0.004						
17	tubeszb	0.003		idea	0.003	exposed
0.003						
18	hours	0.004		video	0.004	testing
0.004						
19	port0	0.003	optional	0.003		faq
0.003						
	11	12	13	14	15	
0	herdsman	0.012	adapter	0.012		mqtt
1	05	0.006	adapter	0.006		config
2	payload	0.006	15	0.006		03
3	starting	0.004	start	0.004		2024
4	2024	0.004	bootloader	0.004		10
5	dec	0.003	state_topic	0.003		cluster
6	2024	0.008	mainthread	0.008		12
7	zigbee	0.004	01	0.004		06
8	node_modules	0.003	wifi	0.003		22
9	activity	0.008	comment	0.008		label
10	developers	0.003	specify	0.003		fss
11	dockerfile	0.003	overlay	0.003		let
12	clicking	0.003	finding	0.003	successful	
13	causes	0.003	branch	0.003		reconnect
14	configurations	0.003	driver	0.003		zzh
15	days	0.003	warn	0.003		opening
16	autoopen	0.004	linkquality	0.003		switch
17	successfully	0.003	skip	0.003		aware
18	stdout	0.004	218	0.003		normal
19	cc2530	0.003	new_api	0.003		ok

En los parámetros de la función `print_topics` se indicó que se mostrarán por pantalla un total de 20 temas con 8 palabras por cada tema. En la primera columna del DataFrame se indica el índice de cada tema, y en las siguientes se muestra primero el porcentaje y seguido de este la palabra a la que pertenece. Por ejemplo, en *Topic 0*(primera fila), vemos que la palabra **zigbee2mqtt** (columna 1) cuenta con una *frecuencia* de **0.029** (columna 0).

Debido a que aún se trata de una versión experimental, no es recomendable el uso exclusivo de este método, pero sí puede servir de ayuda para la comparación de resultados con otros modelos ya asentados.

Teniendo en cuenta que los documentos con los que se suele trabajar no están conformados por un único tema, si no por un conjunto de varios, a veces puede ser complicado utilizar *modelos de temas* para estas tareas. Por ello, quizás una mejor opción sea el *Clustering*.

Clustering para descubrir la estructura de textos

Además del *Modelado de temas*, existen una gran cantidad de métodos no supervisados, pero no todos son adecuados para datos en formato de texto. Sin embargo, hay varios *algoritmos de clustering* que pueden servir para el desarrollo de este tipo de tareas.

Como el dataset con el que se está trabajando está formado por miles de comentarios se entiende que cada uno de estos textos tienen su propio tema (un error de instalación, funciones con errores, etc) que puede ser extraído en base a la estructura del propio texto.

En este caso, se utilizará el algoritmo *k-means clustering* para el desarrollo de la tarea.

K-means clustering

Como ya se vio en el pasado, se sabe que el clustering es un proceso con un consumo de tiempo elevado y hay que tener en cuenta que los procesos pueden llevar incluso horas para completar la tarea, sin embargo, el algoritmo k-means está muy bien optimizado y el tiempo de ejecución se reduce considerablemente.

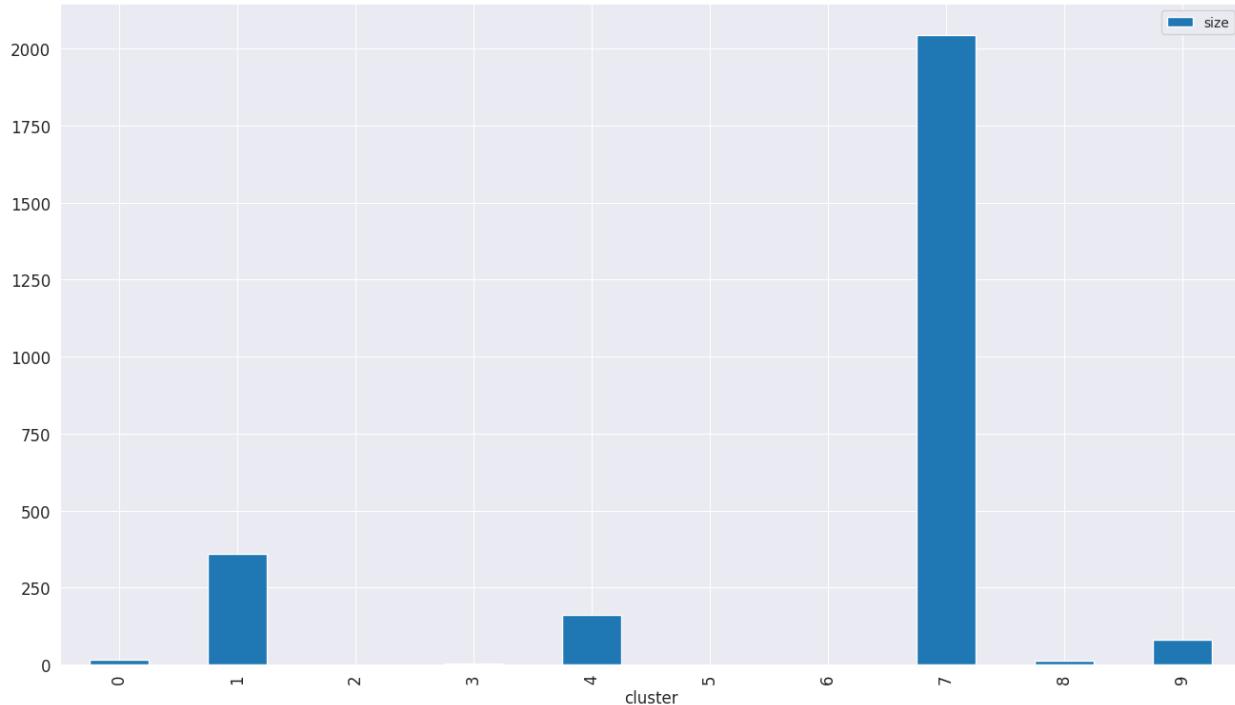
Hay que recordar que el funcionamiento de este algoritmo se basa en agrupar los textos con temas similares en los diferentes clusters. Como norma general, y debido a la heterogeneidad de los comentarios, la mayoría de los clusters tienden a ser relativamente pequeños y compuestos por comentarios que comparten similitudes muy obvias, junto con otro cluster al que se le asignan el resto.

Se volverá a utilizar la librería *scikit-learn* porque su API para clustering es similar a la utilizada para el modelado de temas. En este momento, se hará uso de un total de 10 clusters.

```
from sklearn.cluster import KMeans  
  
k_means_comms = KMeans(n_clusters=10, random_state=42)  
k_means_comms.fit(tfidf_text_vectors)  
  
KMeans(n_clusters=10, random_state=42)
```

Ahora es mucho más sencillo conocer el número de comentarios asignados a cada clúster.

```
sizes = []  
  
for i in range(10):  
    sizes.append({"cluster": i, "size":  
np.sum(k_means_comms.labels_==i)})  
pd.DataFrame(sizes).set_index("cluster").plot.bar(figsize=(16,9))  
  
<Axes: xlabel='cluster'>
```



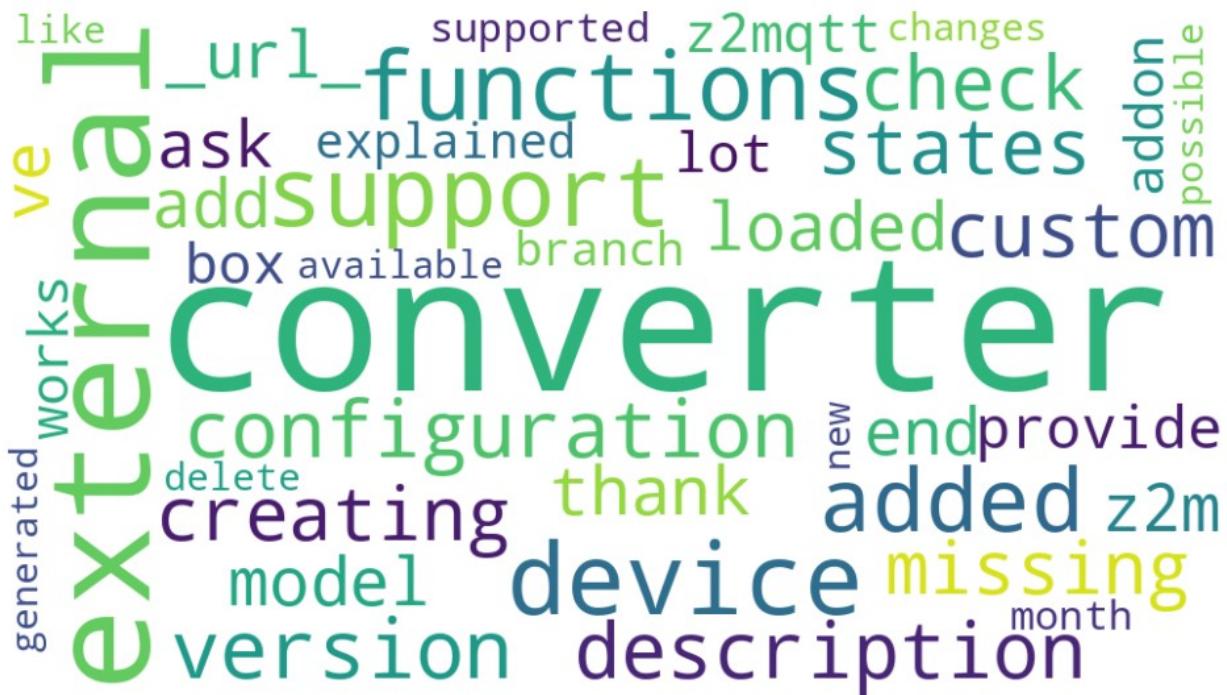
```
np.unique(k_means_comms.labels_, return_counts=True)
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=int32),
 array([ 17,  359,     1,     2,  160,     1,     1, 2044,    12,     81]))
```

Se observa que al cluster 0 pertenecen 17 comentarios, al cluster 2 pertenecen 359, o que al 7 se le han asignado 2.044 comentarios.

Se pueden mostrar por pantalla las nubes de palabras de los clusters para comparar con aquellas generados mediante métodos de modelado de temas.

```
def wordcloud_clusters(model, vectors, features, no_top_words=40):
    for cluster in np.unique(model.labels_):
        size = {}
        words = vectors[model.labels_ == cluster].sum(axis=0).A[0]
        largest = words.argsort()[:-1] # invert sort order
        for i in range(0, no_top_words):
            size[features[largest[i]]] = abs(words[largest[i]])
        wc = WordCloud(background_color="white", max_words=100,
width=960, height=540)
        wc.generate_from_frequencies(size)
        plt.figure(figsize=(12,12))
        plt.imshow(wc, interpolation='bilinear')
        plt.axis("off")
        # if you don't want to save the topic model, comment the next
line
        #plt.savefig(f'cluster{cluster}.png')
```

```
wordcloud_clusters(k_means_comms, tfidf_text_vectors,  
tfidf_text_vectorizer.get_feature_names_out())
```



docker
installed
zigbee2mqtt
thanks

people
time
remember
released
location
updating
software
stay
comment
41
great
definitely
happen
version
free
know
update
automatically
works
think
till
remote

zigbee2mqtt
error
info
config
work
addon
little
link
check
idea

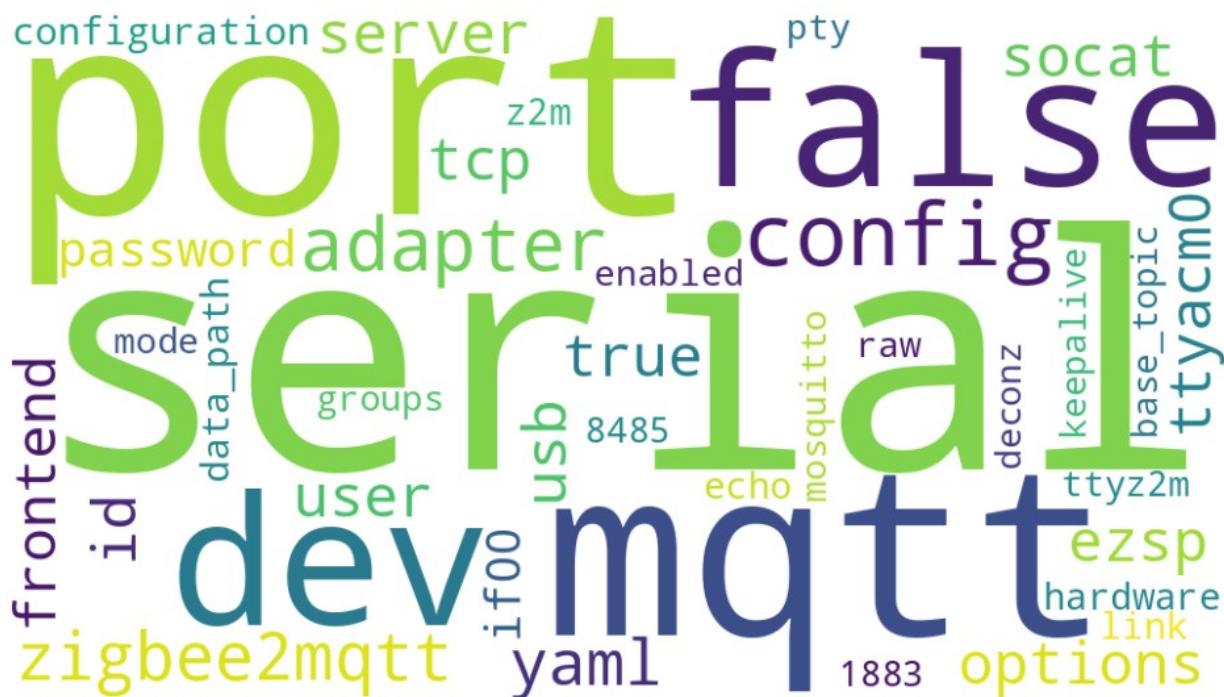
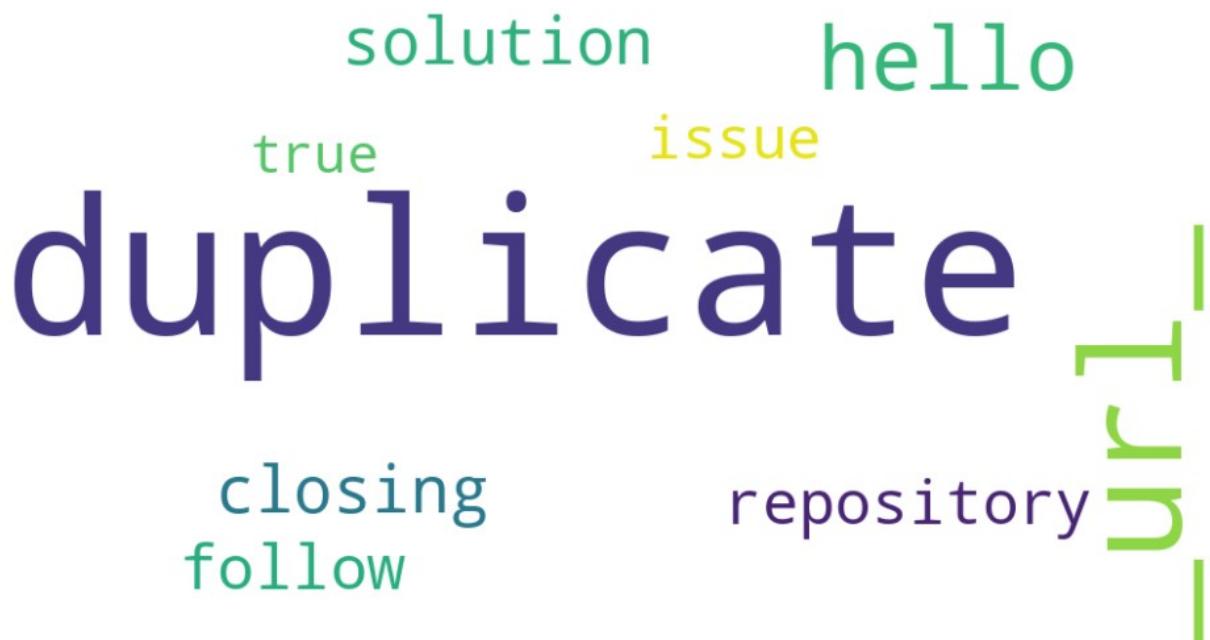
adapter
node_modules
app
npm
failed
zigbee
herdsman
ezsp
znp

null
src
2023
08
05
21
17
11
03
12
04
18
07
ts

log
starting
02
01
2024
2021
lib
start
01
2024
2021
14

list 2024
work help
new
version connecting
38 z2m
devices information logs

version problem work
updated like image
edge new assistant
mqtt ha thank working
try zigbee think don
zigbee2mqtt config supervisor
fixed z2m yaml addon
koenkk works devices
thanks url device home add
fix configuration update latest use



Se puede observar que pese a que en algunos casos los temas no varían demasiado, hay otras nubes de palabras que parecen más concisas respecto a los temas y pueden aportar más información.

```
# Muestra por pantalla las palabras más representativas de cada clúster
```

```

# más fácil la comparación que con las nubes de palabras #

# Obtener los centroides de los clusters
centroids = k_means_comms.cluster_centers_

# Obtener las palabras más representativas de cada cluster
terms = tfidf_text_vectorizer.get_feature_names_out()

num_words = 10 # Número de palabras clave por cluster
top_words = {}

for i, centroid in enumerate(centroids):
    top_indices = centroid.argsort()[-num_words:][::-1] # Obtener los
    # índices de las palabras más relevantes
    top_words[f"Cluster {i}"] = [terms[ind] for ind in top_indices]

# Mostrar los temas por cluster
df_top_words = pd.DataFrame(top_words)

print(df_top_words)

```

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Cluster 5 \	converter	stale	installed	know	error
0 little	external	days	docker	don	info
1 idea	device	activity	thanks	stay	zigbee2mqtt
2 link	support	label	zigbee2mqtt	till	zigbee
3 check	functions	comment	zzh	released	herdsman
4 work	added	closed	error	remote	app
5 new	version	remove	entries	location	adapter
6 addon	configuration	30	entry	people	2022
7 config	description	open	entrypoint	automatically	starting
8 errno	states	issue	enum	41	start
9 error					
	Cluster 6	Cluster 7	Cluster 8	Cluster 9	
0	work	issue	duplicate	serial	
1	devices	_url_	_url_	port	
2	connecting	problem	hello	mqtt	
3	version	thanks	closing	false	
4	information	zigbee2mqtt	solution	dev	

```
5      list          add       follow      config
6      38           z2m      repository   adapter
7      2024          version    issue      zigbee2mqtt
8      help          addon     true       yaml
9      logs          update   environment server
```