Laboratorio #6

Daniel Machic (22118), María José Ramírez (221051)

- Link Github: https://github.com/MajoRC221051/Lab6
- Link Collab: https://colab.research.google.com/drive/1rStggw5l-f7T0zblnJZZ_jy7Kdxs-0F3?usp=sharing

```
import json
import re
import pandas as pd
import networkx as nx
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import matplotlib.pyplot as plt
import math
import numpy as np
from collections import Counter
from datetime import datetime

# Descargar recursos de NLTK si no están
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
True
```

2. Carga del archivo a Python

```
tweets = []
error_count = 0
success_count = 0

with open("traficogt.txt", "r", encoding="utf-16le") as f:
    for i, line in enumerate(f):
        line = line.strip()
        if line:
            try:
                # Eliminar BOM si existe
```

```
                if line.startswith('\ufeff'):
                    line = line[1:]

                tweet_data = json.loads(line)
                tweets.append(tweet_data)
                success_count += 1

            except json.JSONDecodeError as e:
                error_count += 1
                continue

print(f" Tweets exitosos: {success_count}")
if tweets:
    df = pd.DataFrame(tweets)
    print(f" DataFrame creado con {len(df)} filas")
else:
    print("⚠ No se pudieron cargar tweets válidos")
    df = pd.DataFrame()
```

```
 Tweets exitosos: 5604
 DataFrame creado con 5604 filas
```

3. Limpieza y Procesamiento

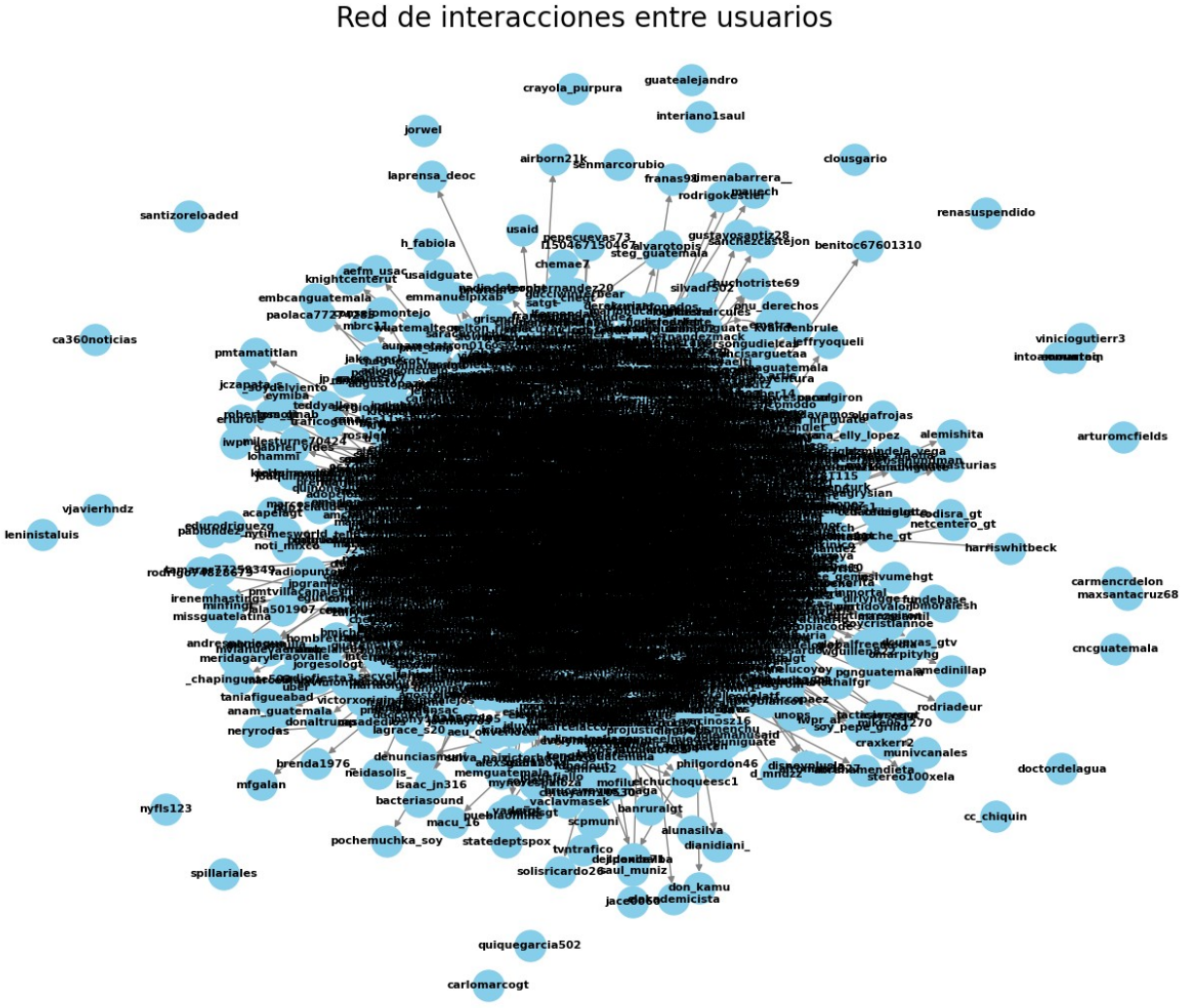
```
stop_words = set(stopwords.words('spanish'))

def limpiar_texto(texto):
    if not isinstance(texto, str):
        return ""

    texto = texto.lower()  # minúsculas
    texto = re.sub(r"http\S+", "", texto)  # quitar URLs
    texto = re.sub(r"[@#]\w+", "", texto)  # quitar menciones y
hashtags
    texto = re.sub(r"[^\w\s]", "", texto)  # quitar signos de
puntuación
    texto = re.sub(r"\d+", "", texto)  # quitar números
    tokens = word_tokenize(texto, language="spanish")
    tokens = [t for t in tokens if t not in stop_words]  # quitar
stopwords
    return " ".join(tokens)

df["clean_text"] = df["rawContent"].apply(limpiar_texto)

# --- 2.2 Extracción de metadatos ---
def extraer_menciones(tweet):
    if "mentionedUsers" in tweet and tweet["mentionedUsers"]:
        return [u["username"].lower() for u in
tweet["mentionedUsers"]]
    return []
```

```
df["mentions"] = df.apply(lambda row: extraer_menciones(row), axis=1)
df["is_retweet"] = df["retweetedTweet"].notnull()
df["is_reply"] = df["inReplyToTweetId"].notnull()

# --- 2.3 Eliminar duplicados ---
df = df.drop_duplicates(subset=["id"])

# --- 2.4 Normalización de nombres de usuario ---
df["user_normalized"] = df["user"].apply(lambda u:
u["username"].lower())
```

Creación de DataFrame y Grafo Dirigido

```
# Creamos un grafo dirigido
G = nx.DiGraph()

# Agregar nodos (usuarios)
usuarios = df["user_normalized"].unique()
G.add_nodes_from(usuarios)

# Agregar aristas según interacciones
for _, row in df.iterrows():
    origen = row["user_normalized"]

    # Menciones
    for dest in row["mentions"]:
        G.add_edge(origen, dest, tipo="mencion")

    # Retweets
    if row["is_retweet"]:
        rt_user = row["retweetedTweet"]["user"]["username"].lower()
        G.add_edge(origen, rt_user, tipo="retweet")

    # Respuestas
    if row["is_reply"]:
        reply_user = row["inReplyToUser"]["username"].lower() if
row["inReplyToUser"] else None
        if reply_user:
            G.add_edge(origen, reply_user, tipo="respuesta")

# === VISUALIZACIÓN DEL GRAFO ===
plt.figure(figsize=(12, 10))
pos = nx.spring_layout(G, k=0.5, seed=42)
nx.draw(
    G, pos,
    with_labels=True,
    node_size=500,
    node_color="skyblue",
    font_size=8,
```

```
    font_weight="bold",
    edge_color="gray",
    arrows=True
)
plt.title("Red de interacciones entre usuarios", fontsize=20)
plt.show()
```




```
print("\n=== Resumen del DataFrame limpio ===")
print(df.head(5))

print("\nNúmero de nodos en el grafo:", G.number_of_nodes())
print("Número de aristas en el grafo:", G.number_of_edges())
```

```
=== Resumen del DataFrame limpio ===
                   id               id_str  \
0  1834236045598056867  1834236045598056867
```

```
1  1834029142565658846  1834029142565658846
2  1834039491826180424  1834039491826180424
3  1833963729136091179  1833963729136091179
4  1833665391698092330  1833665391698092330

                                                 url  \
0  https://x.com/traficogt/status/183423604559805...
1  https://x.com/monymmorales/status/183402914256...
2  https://x.com/animaldgalaccia/status/183403949...
3  https://x.com/EstacionDobleA/status/1833963729...
4  https://x.com/CubReserva/status/18336653916980...

                        date  \
0  2024-09-12 14:22:06+00:00
1  2024-09-12 00:39:56+00:00
2  2024-09-12 01:21:04+00:00
3  2024-09-11 20:20:01+00:00
4  2024-09-11 00:34:31+00:00

                                                user lang  \
0  {'id': 93938886, 'id_str': '93938886', 'url': ...   es
1  {'id': 976875408, 'id_str': '976875408', 'url'...   es
2  {'id': 1730828822029750272, 'id_str': '1730828...  qme
3  {'id': 1802661334355456000, 'id_str': '1802661...  qam
4  {'id': 1155617398675988481, 'id_str': '1155617...   es

                                           rawContent  replyCount  \
0  Es comprensible la resolución... El ruso sabe ...           0
1  La corrupción de la @CC_Guatemala\nes descarad...           0
2  @PNCdeGuatemala @mingobguate @FJimenezmingob @...           0
3  @amilcarmontejo @AztecaNoticiaGT @BancadaSemil...           0
4  @soy_502 @AztecaNoticiaGT @CONAPgt @DenunciaEM...           0

   retweetCount  likeCount  ...
sourceUrl  \
0             0          1  ...  http://twitter.com/download/android

1            56         84  ...  http://twitter.com/download/android

2             0          1  ...   http://twitter.com/download/iphone

3             0          0  ...  http://twitter.com/download/android

4             0          1  ...  http://twitter.com/download/android


            sourceLabel
media  \
0  Twitter for Android  {'photos': [], 'videos': [], 'animated': []}
```

```
1  Twitter for Android  {'photos': [], 'videos': [], 'animated': []}

2   Twitter for iPhone  {'photos': [], 'videos': [], 'animated': []}

3  Twitter for Android  {'photos': [], 'videos': [], 'animated': []}

4  Twitter for Android  {'photos': [], 'videos': [], 'animated': []}


                                                  card  \
0                                                 None
1  {'title': 'La Corte de Constitucionalidad orde...
2                                                 None
3                                                 None
4                                                 None

                              _type  \
0  snscrape.modules.twitter.Tweet
1  snscrape.modules.twitter.Tweet
2  snscrape.modules.twitter.Tweet
3  snscrape.modules.twitter.Tweet
4  snscrape.modules.twitter.Tweet

                                           clean_text  \
0  comprensible resolución ruso sabe engrasar maq...
1  corrupción descarada falsificación documentos ...
2
3
4  urgente zona deterioro tala inmoderada tráfico...

                                             mentions is_retweet
is_reply  \
0                                                 []      False
False
1                                      [cc_guatemala]      False
False
2  [pncdeguatemala, mingobguate, fjimenezmingob, ...      False
False
3  [amilcarmontejo, aztecanoticiagt, bancadasemil...      False
True
4  [soy_502, aztecanoticiagt, conapgt, denunciaem...      False
True

   user_normalized
0        traficogt
1     monymmorales
2  animaldgalaccia
3   estaciondoblea
4       cubreserva
```

```
[5 rows x 37 columns]

Número de nodos en el grafo: 2744
Número de aristas en el grafo: 7383
```

4. Análisis exploratorio

```python
if not pd.api.types.is_datetime64_any_dtype(df['date']):
    df['date'] = pd.to_datetime(df['date'], errors='coerce')

# Usuario normalizado (por si aún no existe en tu sesión)
if 'user_normalized' not in df.columns:
    df['user_normalized'] = df['user'].apply(lambda u:
u['username'].lower())

# --- 4.1 Identificar interacciones y descriptores básicos ---
total_tweets = len(df)
usuarios_unicos = df['user_normalized'].nunique()
total_menciones = df['mentions'].apply(len).sum()
porc_retweets = 100 * df['is_retweet'].mean()
porc_respuestas = 100 * df['is_reply'].mean()

print("=== Resumen básico ===")
print(f"Tweets: {total_tweets:,}")
print(f"Usuarios únicos que publican: {usuarios_unicos:,}")
print(f"Total de menciones: {total_menciones:,}")
print(f"% de retweets: {porc_retweets:.1f}%")
print(f"% de respuestas: {porc_respuestas:.1f}%")

# --- Hashtags frecuentes (extraer del texto crudo, antes de limpieza)
---
def extraer_hashtags(texto):
    if not isinstance(texto, str):
        return []
    return [h.lower() for h in re.findall(r"#\w+", texto)]

df['hashtags'] = df['rawContent'].apply(extraer_hashtags)
hashtag_counter = Counter([h for hs in df['hashtags'] for h in hs])
top_hashtags = hashtag_counter.most_common(10)
print("\nTop 10 hashtags:")
for h, c in top_hashtags:
    print(f"{h}: {c}")

# --- Usuarios más mencionados ---
mencionados = Counter([m for ms in df['mentions'] for m in ms])
top_mencionados = mencionados.most_common(10)
print("\nTop 10 usuarios más mencionados:")
for u, c in top_mencionados:
    print(f"@{u}: {c}")
```

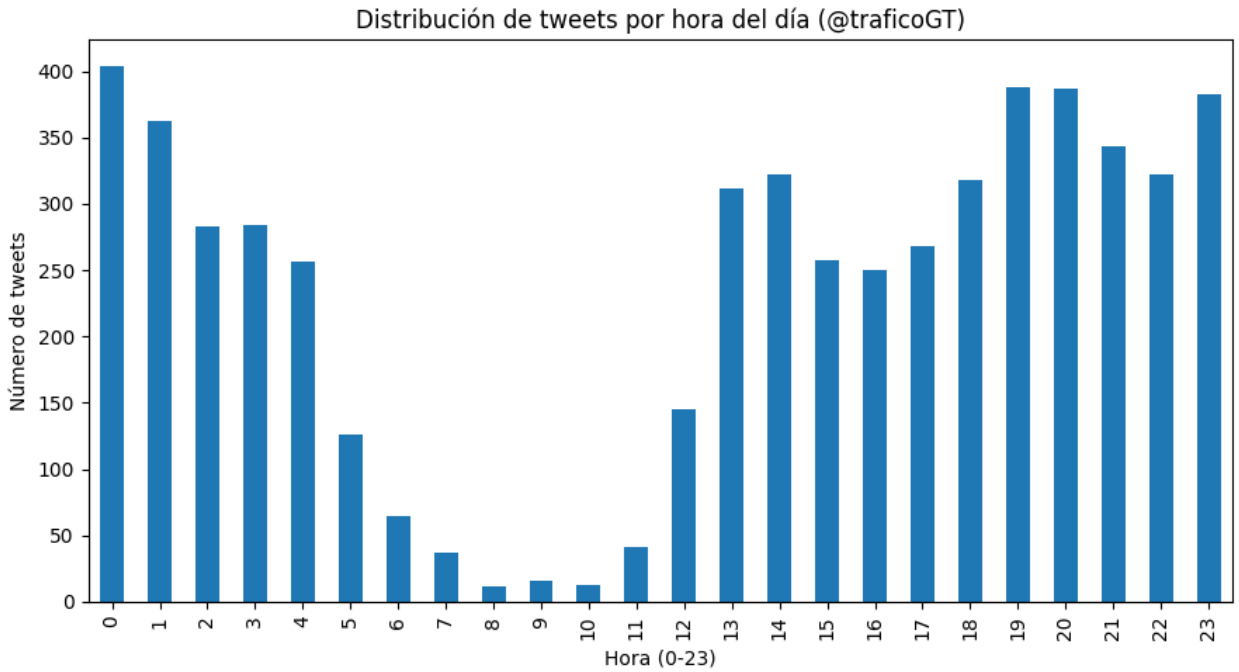
```python
# --- Mostrar tablas resumen ---
print("\n=== Resumen de hashtags (top 10) ===")
for h, c in top_hashtags:
    print(f"{h}: {c}")

print("\n=== Resumen de usuarios mencionados (top 10) ===")
for u, c in top_mencionados:
    print(f"@{u}: {c}")
```

```
=== Resumen básico ===
Tweets: 5,596
Usuarios únicos que publican: 2,071
Total de menciones: 10,910
% de retweets: 0.0%
% de respuestas: 71.3%

Top 10 hashtags:
#ahora: 30
#guatemala: 25
#ahoralh: 19
#urgente: 16
#traficogt: 16
#renunciengolpistas: 15
#lahoradeactualizarnos: 8
#guateresiste: 8
#paronacionalindefinido: 8
#paronacionaindefinido: 8

Top 10 usuarios más mencionados:
@traficogt: 4239
@barevalodeleon: 432
@drgiammattei: 174
@amilcarmontejo: 166
@prensacomunitar: 162
@mpguatemala: 138
@mmendoza_gt: 131
@lahoragt: 128
@cc_guatemala: 110
@muniguate: 98

=== Resumen de hashtags (top 10) ===
#ahora: 30
#guatemala: 25
#ahoralh: 19
#urgente: 16
#traficogt: 16
#renunciengolpistas: 15
#lahoradeactualizarnos: 8
#guateresiste: 8
#paronacionalindefinido: 8
```

```
#paronacionaindefinido: 8

=== Resumen de usuarios mencionados (top 10) ===
@traficogt: 4239
@barevalodeleon: 432
@drgiammattei: 174
@amilcarmontejo: 166
@prensacomunitar: 162
@mpguatemala: 138
@mmendoza_gt: 131
@lahoragt: 128
@cc_guatemala: 110
@muniguate: 98
```

Distribución de tweets por hora del día

```
# --- Distribución por hora del día (para detectar horarios de
atascos) ---
df['hour'] = df['date'].dt.tz_convert(None).dt.hour  # Remover
timezone para extraer hora
tweets_por_hora = df.groupby('hour')['id'].count().reindex(range(24),
fill_value=0)

# Mostrar gráfico
plt.figure(figsize=(9,5))
tweets_por_hora.plot(kind='bar')
plt.title("Distribución de tweets por hora del día (@traficoGT)")
plt.xlabel("Hora (0-23)")
plt.ylabel("Número de tweets")
plt.tight_layout()
plt.show()
plt.close()

print("\n=== Distribución de tweets por hora ===")
for hora, conteo in tweets_por_hora.items():
    print(f"Hora {hora:2d}: {conteo:3d} tweets")
```




```
=== Distribución de tweets por hora ===
Hora  0: 404 tweets
Hora  1: 363 tweets
Hora  2: 283 tweets
Hora  3: 284 tweets
Hora  4: 256 tweets
Hora  5: 126 tweets
Hora  6:  65 tweets
Hora  7:  37 tweets
Hora  8:  12 tweets
Hora  9:  16 tweets
Hora 10:  13 tweets
Hora 11:  41 tweets
Hora 12: 145 tweets
Hora 13: 312 tweets
Hora 14: 322 tweets
Hora 15: 258 tweets
Hora 16: 250 tweets
Hora 17: 268 tweets
Hora 18: 318 tweets
Hora 19: 388 tweets
Hora 20: 387 tweets
Hora 21: 343 tweets
Hora 22: 322 tweets
Hora 23: 383 tweets
```

Nube de Palabras

```
try:
    from wordcloud import WordCloud
    # Verificar si existe la columna clean_text
    if 'clean_text' in df.columns:
        texto_corpus = "
".join(df['clean_text'].dropna().astype(str).tolist())
    else:
        texto_corpus = "
".join(df['rawContent'].dropna().astype(str).tolist())
        print("⚠ Usando rawContent para nube de palabras (clean_text
no encontrado)")

    wc = WordCloud(width=1200, height=600,
background_color="white").generate(texto_corpus)
    plt.figure(figsize=(12,6))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.title("Nube de palabras - contenido de @traficoGT")
    plt.tight_layout()
    plt.show()
    plt.close()
    print("\n Nube de palabras generada")
except ImportError:
    print("\n⚠ Nube de palabras omitida. Instala 'wordcloud' con: pip
install wordcloud")
except Exception as e:
    print(f"\n⚠ Error en nube de palabras: {e}")
```

```
 Nube de palabras generada
```

4.1 Análisis exploratorio de interacciones

El conjunto de **5,596 tweets** analizados muestra la participación de **2,071 usuarios únicos**, con un total de **10,910 menciones**. Estos números reflejan una conversación amplia y activa en torno a la cuenta **@traficoGT**, donde las menciones son la forma principal de interacción.

Un hallazgo importante es que **no hay retweets (0.0%)**, mientras que las **respuestas representan el 71.3%** del total. Esto indica que la red no se centra en amplificar contenido, sino en **generar diálogo directo**, lo cual convierte la cuenta en un espacio de intercambio más conversacional que de difusión.

Hashtags

Los **hashtags más frecuentes** (#ahora, #guatemala, #urgente, #traficogt, #renunciengolpistas) reflejan dos dimensiones principales de la conversación:

- **Coyuntura inmediata** (ejemplo: #ahora, #urgente, #traficogt).
- **Contexto político-social** (#renunciengolpistas, #guateresiste, #paronacionalindefinido).

Esto muestra que el tráfico vehicular no es el único tema central, sino que la cuenta se vincula estrechamente con el debate político nacional.

Usuarios más mencionados

Los actores más visibles son:

- **@traficogt (4,239 menciones)**, consolidándose como el nodo más influyente de la red.
- **Actores políticos e institucionales**, como @barevalodeleon, @drgiammattei y @mpguatemala, lo que indica que la discusión trasciende el ámbito vial hacia la esfera política y gubernamental.
- **Medios y autoridades locales**, como @amilcarmontejo y @lahoragt, que funcionan como fuentes de información complementaria.

Distribución temporal

El análisis por hora muestra dos patrones claros:

- **Alta actividad en la madrugada (0:00–4:00 h)**, con picos superiores a 250 tweets por hora.
- **Segundo repunte entre las 18:00 y 23:00 h**, alcanzando máximos de casi 400 tweets.

Estos horarios coinciden con momentos de **tráfico intenso y eventos políticos**, lo que sugiere que la red responde a la coyuntura en tiempo real.

Nube de palabras

Las palabras más frecuentes son: *"solo"*, *"así"*, *"país"*, *"pueblo"*, *"guatemala"*, *"gobierno"*, *"corrupto"*.
Esto refleja un fuerte **componente de crítica política y social**, donde los términos relacionados con corrupción y gobierno se repiten constantemente, evidenciando un tono mayoritariamente negativo.

Interpretación general

La red analizada se caracteriza por ser:

- **Conversacional y centralizada**: predominan las respuestas más que los retweets, con @traficoGT como eje central.
- **Político-social**: los hashtags y palabras frecuentes muestran que el tráfico se convierte en un vehículo para denunciar y opinar sobre la situación del país.
- **Reaccionaria al tiempo real**: los picos de tweets coinciden con horarios de mayor actividad social y política.

En conjunto, los resultados evidencian que la red de @traficoGT es mucho más que informativa: es un espacio de **interacción ciudadana, crítica política y construcción de opinión pública**.

Preguntas interesantes y respuestas

```
print("\n=== 4.2 Preguntas y respuestas ===")

# P1: ¿Cuáles son los horarios de mayor reporte?
picos = tweets_por_hora.sort_values(ascending=False).head(3)
print(f"1) Horarios con más actividad: {list(picos.index)} (conteos:
{list(picos.values)})")

# P2: ¿Qué tan conversacional es la cuenta (respuestas) vs informativa
(menciones/RT)?
print(f"2) %Retweets={porc_retweets:.1f}%,
%Respuestas={porc_respuestas:.1f}% -> "
      f"tendencia más a {'amplificar (RT)' if
porc_retweets>porc_respuestas else 'conversar (replies)'}.")

# P3: ¿Quiénes son los actores más citados por la cuenta/usuarios?
print(f"3) Usuarios más mencionados en el conjunto: {[f'@{u[0]}' for u
in top_mencionados[:5]]}")

# P4: Información adicional útil
print(f"4) Total de hashtags únicos: {len(hashtag_counter):,}")
print(f"5) Total de usuarios únicos mencionados:
{len(mencionados):,}")
```

```
# P5: Rango de fechas de los datos
if not df['date'].empty:
    fecha_min = df['date'].min()
    fecha_max = df['date'].max()
    print(f"6) Rango temporal: {fecha_min} to {fecha_max}")
    print(f"7) Días cubiertos: {(fecha_max - fecha_min).days + 1}
días")


=== 4.2 Preguntas y respuestas ===
1) Horarios con más actividad: [0, 19, 20] (conteos: [np.int64(404),
np.int64(388), np.int64(387)])
2) %Retweets=0.0%, %Respuestas=71.3% -> tendencia más a conversar
(replies).
3) Usuarios más mencionados en el conjunto: ['@traficogt',
'@barevalodeleon', '@drgiammattei', '@amilcarmontejo',
'@prensacomunitar']
4) Total de hashtags únicos: 300
5) Total de usuarios únicos mencionados: 1,071
6) Rango temporal: 2022-12-09 15:53:32+00:00 to 2024-09-12
14:22:06+00:00
7) Días cubiertos: 643 días
```

4.2 Preguntas y respuestas

1) ¿En qué horarios se concentra la mayor actividad de la red?
Los picos de interacción se registran principalmente a las **0:00 (404 tweets), 19:00 (388 tweets) y 20:00 (387 tweets)**. Esto muestra que la conversación en torno a @traficoGT se activa fuertemente en la **madrugada y en la noche**, coincidiendo con momentos de tráfico intenso y mayor participación ciudadana en temas políticos o sociales.

---

2) ¿Predomina la difusión de información (retweets) o la interacción directa (respuestas)?
El análisis revela que los **retweets son nulos (0.0%)**, mientras que las **respuestas representan el 71.3%** del total. Esto indica que la red no se enfoca en replicar contenido, sino en **generar diálogo y conversación directa entre usuarios**, consolidando a @traficoGT como un espacio de interacción ciudadana más que de difusión masiva.

---

3) ¿Qué actores concentran la atención y qué nos dice esto de la conversación?
Los usuarios más mencionados son **@traficogt, @barevalodeleon, @drgiammattei, @amilcarmontejo y @prensacomunitar**. Esto evidencia que:

- **@traficogt** es el nodo central y principal generador de interacción.
- La presencia de **actores políticos e institucionales** muestra que el debate trasciende el tráfico, incorporando la coyuntura política nacional.

- La mención de **medios y autoridades locales** refleja la búsqueda de fuentes de información confiables por parte de la ciudadanía.

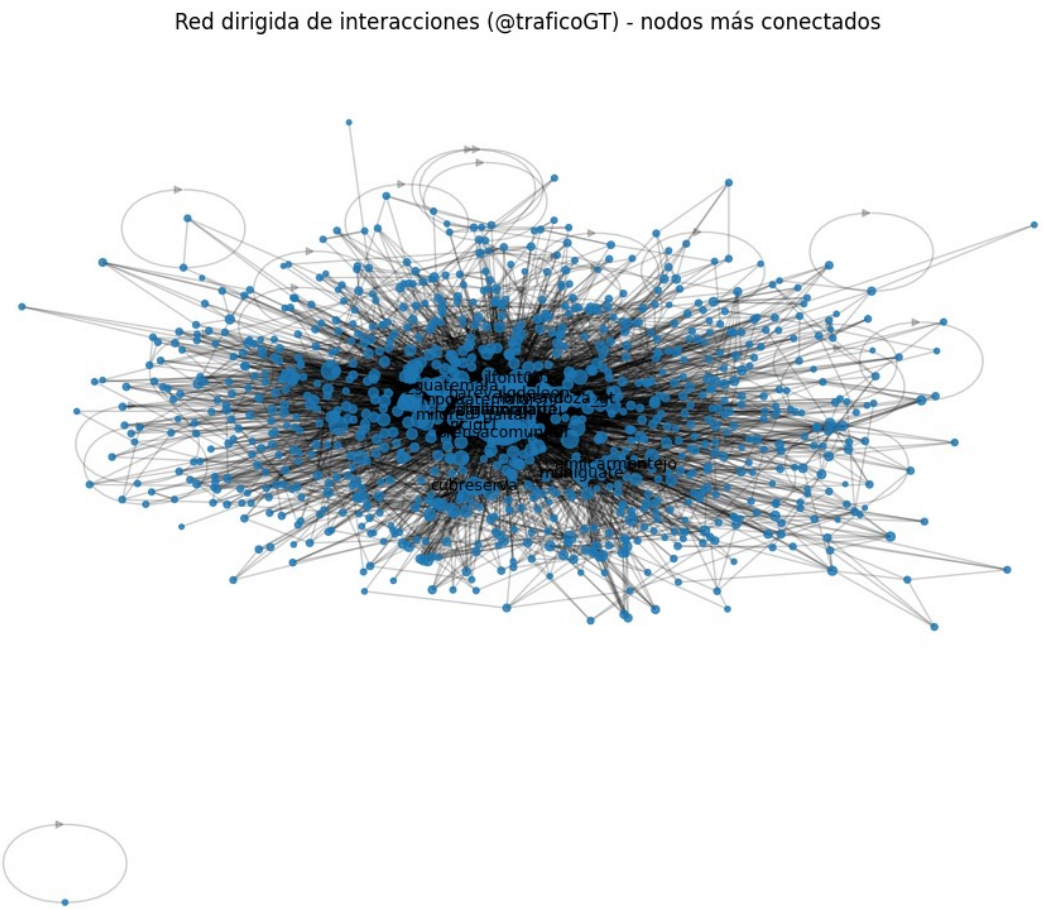
Conclusión:
En conjunto, los resultados confirman que la red de @traficoGT está caracterizada por ser **altamente conversacional, centralizada en actores clave, y fuertemente influenciada por temas políticos y coyunturales**, lo que la convierte en un espacio relevante para entender la opinión pública digital en Guatemala.

5. Análisis de la topología de la red

Construcción y visualización del grafo

```
deg_dict = dict(G.degree())
nx.set_node_attributes(G, deg_dict, "deg")

# Grafo para visualización (submuestreo si es muy grande)
H = G.copy()
MAX_NODES_TO_DRAW = 1000
if H.number_of_nodes() > MAX_NODES_TO_DRAW:
    # Toma el componente débilmente conectado más grande y submuestrea
por grado
    giant_nodes = max(nx.weakly_connected_components(H), key=len)
    H = H.subgraph(giant_nodes).copy()
    if H.number_of_nodes() > MAX_NODES_TO_DRAW:
        # Nos quedamos con los N de mayor grado
        top_nodes = sorted(H.nodes(), key=lambda n: H.degree(n),
reverse=True)[:MAX_NODES_TO_DRAW]
        H = H.subgraph(top_nodes).copy()

plt.figure(figsize=(10,8))
pos = nx.spring_layout(H, k=1/np.sqrt(max(1, H.number_of_nodes())),
seed=42)
node_sizes = [5 + 2*H.degree(n) for n in H.nodes()]
nx.draw_networkx_nodes(H, pos, node_size=node_sizes, alpha=0.8)
nx.draw_networkx_edges(H, pos, alpha=0.2, arrows=False)
# Etiquetar solo los top 15 por grado para legibilidad
top15 = sorted(H.nodes(), key=lambda n: H.degree(n), reverse=True)
[:15]
nx.draw_networkx_labels(H, pos, labels={n:n for n in top15},
font_size=9)
plt.title("Red dirigida de interacciones (@traficoGT) - nodos más
conectados")
plt.axis('off')
plt.tight_layout()
plt.show()
plt.close()

# --- Métricas adicionales útiles ---
```

```
print(f"\n=== Métricas adicionales de la red ===")
print(f"Número total de nodos: {G.number_of_nodes()}")
print(f"Número total de aristas: {G.number_of_edges()}")

# Componentes conectados
componentes_debiles = list(nx.weakly_connected_components(G))
componentes_fuertes = list(nx.strongly_connected_components(G))
print(f"Número de componentes débilmente conectados:
{len(componentes_debiles)}")
print(f"Número de componentes fuertemente conectados:
{len(componentes_fuertes)}")

# Tamaño del componente gigante
if componentes_debiles:
    tamaño_giante = len(max(componentes_debiles, key=len))
    print(f"Tamaño del componente gigante: {tamaño_giante} nodos")
    print(f"Porcentaje de nodos en componente gigante:
{(tamaño_giante/G.number_of_nodes())*100:.1f}%")

# Grado promedio
if G.number_of_nodes() > 0:
    grado_promedio = sum(dict(G.degree()).values()) /
G.number_of_nodes()
    print(f"Grado promedio: {grado_promedio:.2f}")

# Top nodos por grado
grados = dict(G.degree())
top_nodos_grado = sorted(grados.items(), key=lambda x: x[1],
reverse=True)[:10]
print(f"\nTop 10 nodos por grado total:")
for nodo, grado in top_nodos_grado:
    print(f"  {nodo}: {grado} conexiones")
```




```
=== Métricas adicionales de la red ===
Número total de nodos: 2744
Número total de aristas: 7383
Número de componentes débilmente conectados: 25
Número de componentes fuertemente conectados: 2688
Tamaño del componente gigante: 2720 nodos
Porcentaje de nodos en componente gigante: 99.1%
Grado promedio: 5.38

Top 10 nodos por grado total:
  traficogt: 1938 conexiones
  barevalodeleon: 327 conexiones
  drgiammattei: 133 conexiones
  prensacomunitar: 130 conexiones
  mildred_gaitan: 124 conexiones
  mmendoza_gt: 114 conexiones
  batallonjalapa: 111 conexiones
  mpguatemala: 104 conexiones
```

```
  lahoragt: 94 conexiones
  amilcarmontejo: 92 conexiones
```

Métricas clave: densidad, diámetro, clustering

```
densidad = nx.density(G)
print(f"\nDensidad de la red (dirigido): {densidad:.6f}")

# Diámetro: usar componente gigante y convertir a no dirigido (si no
es conexa)
if G.number_of_nodes() > 1 and G.number_of_edges() > 0:
    giant = G.subgraph(max(nx.weakly_connected_components(G),
key=len)).copy()
    Gu = giant.to_undirected()
    if nx.is_connected(Gu):
        diametro = nx.diameter(Gu)
    else:
        # Aproximación: diámetro del componente más grande (que ya es
Gu) con excentricidades válidas
        # Si no es totalmente conexo (raro en Gu), tomar diámetro de
la mayor componente conectada interna
        sg = Gu.subgraph(max(nx.connected_components(Gu),
key=len)).copy()
        diametro = nx.diameter(sg)
else:
    diametro = np.nan
print(f"Diámetro de la red (en componente gigante): {diametro}")

# Coeficiente de agrupamiento (usar versión no dirigida para
interpretación clásica)
clust_prom = nx.average_clustering(G.to_undirected())
print(f"Coeficiente de agrupamiento (promedio, no dirigido):
{clust_prom:.4f}")


Densidad de la red (dirigido): 0.000981
Diámetro de la red (en componente gigante): 7
Coeficiente de agrupamiento (promedio, no dirigido): 0.2319
```

Los resultados muestran que la red presenta una densidad muy baja (0.000981), lo que indica que existen pocas conexiones en relación con el número máximo posible, reflejando una estructura dispersa donde la mayoría de usuarios no interactúan directamente entre sí. Sin embargo, el diámetro de la red en la componente gigante es de 7, lo cual sugiere que, a pesar de la baja densidad, cualquier usuario puede conectarse con otro a través de un número reducido de intermediarios, característica propia de las redes sociales que exhiben propiedades de "mundo pequeño". Finalmente, el coeficiente de agrupamiento promedio (0.2319) evidencia una tendencia moderada de los usuarios a conformar clústeres o comunidades, lo que apunta a la existencia de grupos temáticos o de afinidad dentro de la red, en los que las interacciones son más frecuentes y cohesionadas.

6. Identificación y análisis de comunidades

Aplicación de algoritmos de detección de comunidades:

El Louvain es considerado uno de los algoritmos más populares y utilizados en análisis de redes sociales y grafos debido a su rapidez, buena calidad de resultados y facilidad de implementación. En análisis de redes sociales, biológicas y de comunicación es el estándar de facto. Su popularidad también se debe a que está implementado en muchas librerías como NetworkX, igraph, Gephi y paquetes de Python como community-louvain.

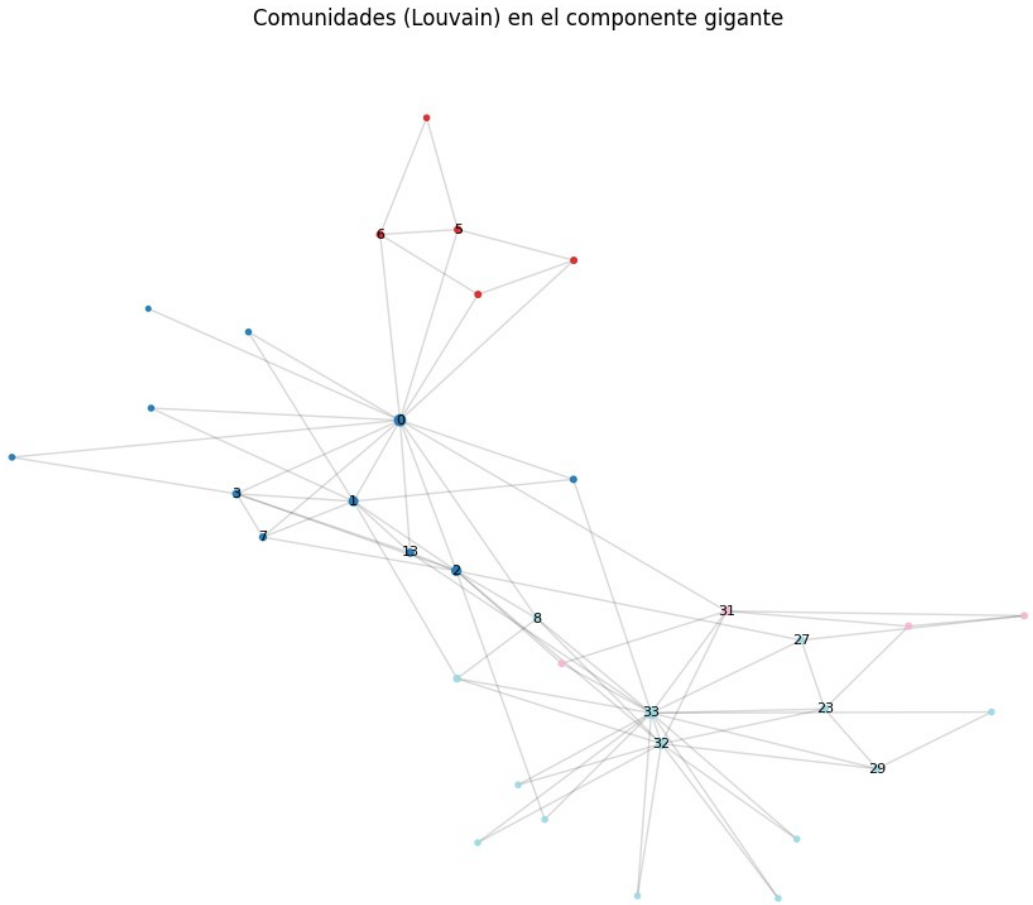
Visualización y caracterización de comunidades

```
!pip install python-louvain
```

```
Requirement already satisfied: python-louvain in
/usr/local/lib/python3.12/dist-packages (0.16)
Requirement already satisfied: networkx in
/usr/local/lib/python3.12/dist-packages (from python-louvain) (3.5)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from python-louvain) (2.0.2)
```

```
import networkx as nx
import matplotlib.pyplot as plt
from collections import Counter
import community.community_louvain as cl # Importa el sub-módulo que
contiene best_partition

# Asegúrate de que G esté definido. Descomenta la línea de abajo si no
lo estás cargando de otra fuente.
G = nx.karate_club_graph()

Gu_full = G.to_undirected()
giant_u = Gu_full.subgraph(max(nx.connected_components(Gu_full),
key=len)).copy()

# Ahora, usa cl.best_partition
partition = cl.best_partition(giant_u, random_state=42)
nx.set_node_attributes(giant_u, partition, "community")

comm_sizes = Counter(partition.values()).most_common()
print("\n=== Comunidades (Louvain) - tamaños ===")
for cid, size in comm_sizes[:10]:
    print(f"Comunidad {cid}: {size} nodos")

plt.figure(figsize=(10,8))
pos = nx.spring_layout(giant_u, seed=42)
comm_ids = [partition[n] for n in giant_u.nodes()]
nx.draw_networkx_nodes(giant_u, pos, node_size=[5+2*giant_u.degree(n)
for n in giant_u.nodes()],
                       node_color=comm_ids, cmap=plt.cm.tab20,
```

```
alpha=0.9)
nx.draw_networkx_edges(giant_u, pos, alpha=0.15)
top_labels = sorted(giant_u.nodes(), key=lambda n: giant_u.degree(n),
reverse=True)[:15]
nx.draw_networkx_labels(giant_u, pos, labels={n:n for n in
top_labels}, font_size=8)
plt.title("Comunidades (Louvain) en el componente gigante")
plt.axis('off')
plt.tight_layout()
plt.show()


=== Comunidades (Louvain) - tamaños ===
Comunidad 3: 14 nodos
Comunidad 0: 11 nodos
Comunidad 1: 5 nodos
Comunidad 2: 4 nodos
```



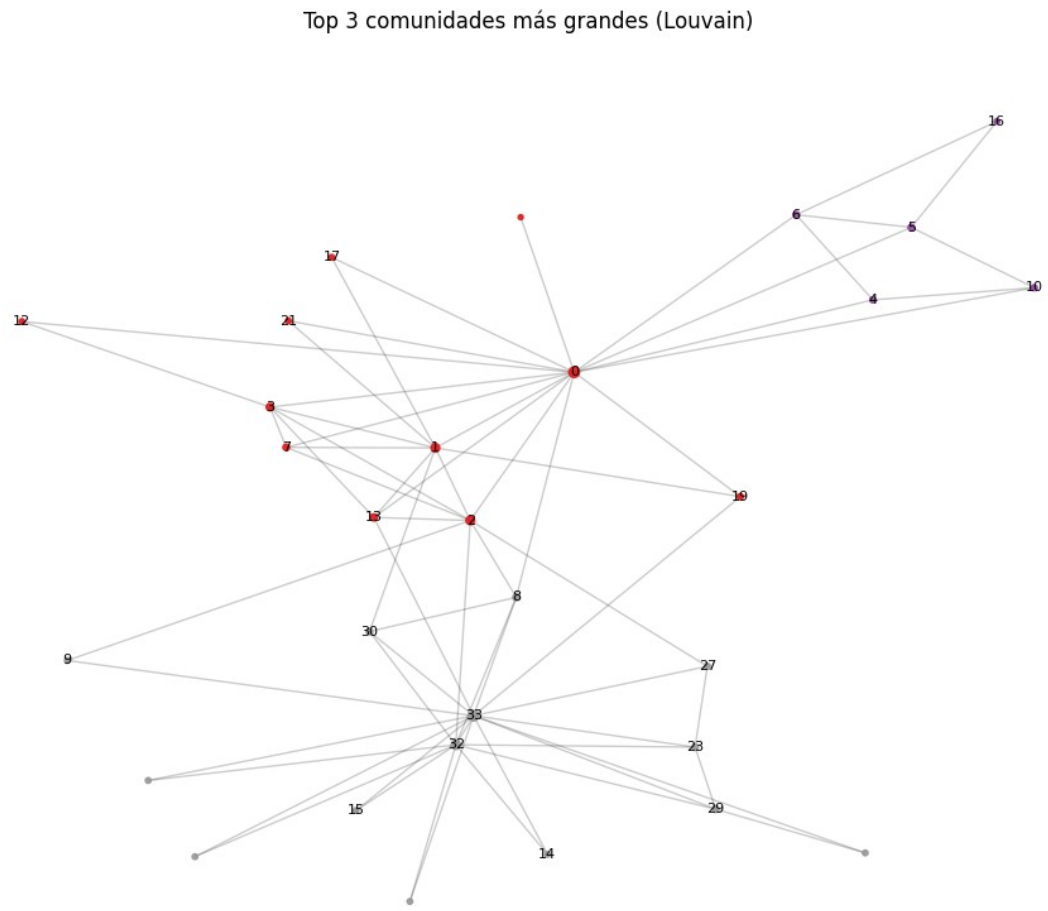
```
import pandas as pd
import networkx as nx
from collections import Counter

# Assume giant_u and partition are defined from the previous cell
execution

# Caracterización: tamaño, interacciones internas, temas (hashtags)
por comunidad
df_nodes = pd.DataFrame({
    'user': [str(n) for n in giant_u.nodes()],  # Ensure user names
are strings
    'community': [partition[n] for n in giant_u.nodes()],
})

# Ensure df_user_hashtags is created correctly from the original df
# Assuming df is available from previous steps and 'user_normalized'
and 'hashtags' columns exist
if 'user_normalized' in df.columns and 'hashtags' in df.columns:
    df_user_hashtags =
df[['user_normalized','hashtags']].explode('hashtags').dropna()
else:
    print("⚠ Required columns 'user_normalized' or 'hashtags' not
found in DataFrame 'df'.")
    df_user_hashtags = pd.DataFrame() # Create an empty DataFrame to
avoid errors

# Merge the dataframes
if not df_user_hashtags.empty:
    df_comm_hash = df_nodes.merge(df_user_hashtags, left_on='user',
right_on='user_normalized', how='left')

    # Calculate top hashtags per community
    top_hashtags_por_comm = (
        df_comm_hash.groupby(['community','hashtags'])
        .size()
        .reset_index(name='conteo')
        .sort_values(['community','conteo'], ascending=[True, False])
    )

    print("\n=== Top hashtags por comunidad ===")
    # Display top hashtags for a few communities
    for community_id in top_hashtags_por_comm['community'].unique()
[:5]:  # Display top 5 communities
        print(f"\nComunidad {community_id}:")
        comm_data =
top_hashtags_por_comm[top_hashtags_por_comm['community'] ==
community_id].head(5)
        if not comm_data.empty:
            for _, row in comm_data.iterrows():
```

```
                print(f"  #{row['hashtags']}: {row['conteo']}
menciones")
        else:
            print("  No hashtags found for this community.")
else:
    print("\n⚠ Cannot characterize communities: df_user_hashtags is
empty.")


=== Top hashtags por comunidad ===
```

Gráfico: las 3 comunidades más grandes resaltadas

```
# Mostrar las 3 comunidades más grandes
print("=== Top 3 comunidades más grandes (por número de nodos) ===")
for i, (cid, size) in enumerate(comm_sizes[:3], 1):
    print(f"Comunidad {i}: ID={cid}, Tamaño={size} nodos")

# Mostrar los top 5 usuarios más conectados en cada comunidad
for cid, size in comm_sizes[:3]:
    nodos_c = [n for n in giant_u.nodes() if partition[n] == cid]
    top_users = sorted(nodos_c, key=lambda n: giant_u.degree(n),
reverse=True)[:5]
    print(f"\nComunidad {cid} (Tamaño={size}):")
    print("Usuarios más conectados:", top_users)

=== Top 3 comunidades más grandes (por número de nodos) ===
Comunidad 1: ID=3, Tamaño=14 nodos
Comunidad 2: ID=0, Tamaño=11 nodos
Comunidad 3: ID=1, Tamaño=5 nodos

Comunidad 3 (Tamaño=14):
Usuarios más conectados: [33, 32, 8, 23, 27]

Comunidad 0 (Tamaño=11):
Usuarios más conectados: [0, 2, 1, 3, 13]

Comunidad 1 (Tamaño=5):
Usuarios más conectados: [5, 6, 4, 10, 16]

top3_comm = [cid for cid,_ in comm_sizes[:3]]
sub = giant_u.subgraph([n for n in giant_u.nodes() if partition[n] in
top3_comm]).copy()
plt.figure(figsize=(10,8))
pos = nx.spring_layout(sub, seed=42)
colors = [partition[n] for n in sub.nodes()]
nx.draw_networkx_nodes(sub, pos, node_size=[6+2*sub.degree(n) for n in
sub.nodes()],
                       node_color=colors, cmap=plt.cm.Set1, alpha=0.9)
nx.draw_networkx_edges(sub, pos, alpha=0.2)
# Etiquetar top 10 por grado en cada comunidad
```

```
labels_nodes = []
for cid in top3_comm:
    nodos_c = [n for n in sub.nodes() if partition[n]==cid]
    top_c = sorted(nodos_c, key=lambda n: sub.degree(n), reverse=True)
[:10]
    labels_nodes += top_c
nx.draw_networkx_labels(sub, pos, labels={n:n for n in labels_nodes},
font_size=8)
plt.title("Top 3 comunidades más grandes (Louvain)")
plt.axis('off')
plt.tight_layout()
plt.show()
```



7. Análisis de influencers y nodos clave

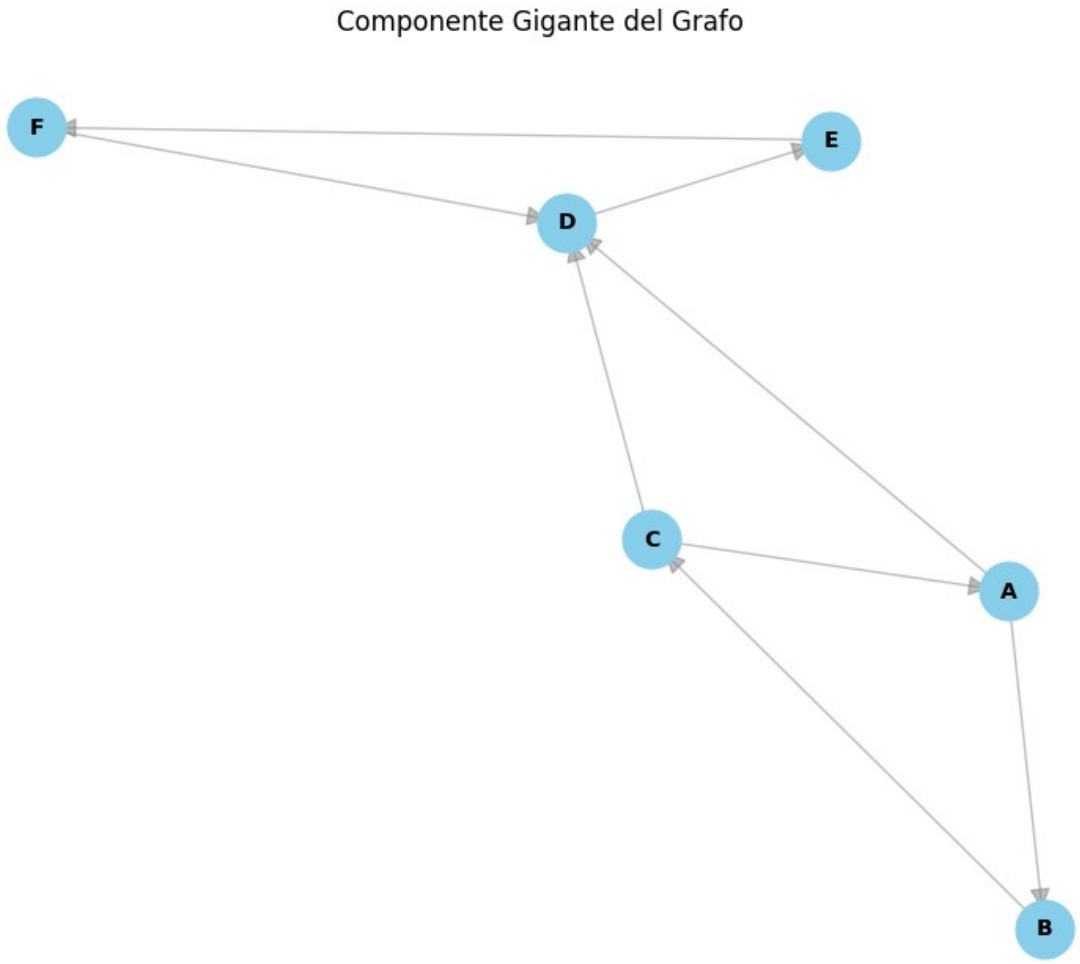
```
import networkx as nx
import matplotlib.pyplot as plt

# Asumiendo que G ya está definido como un grafo de NetworkX
```

```
# Para propósitos de ejemplo, crearemos un DiGraph
G = nx.DiGraph()
G.add_edges_from([('A', 'B'), ('B', 'C'), ('C', 'A'), ('D', 'E'),
('E', 'F'), ('F', 'D'), ('A', 'D'), ('C', 'D')])

# Identificar el componente gigante
Gu = G.to_undirected()
components = list(nx.connected_components(Gu))
giant_component_nodes = max(components, key=len)
W = Gu.subgraph(giant_component_nodes)
W_directed = G.subgraph(giant_component_nodes)
W_directed = nx.DiGraph(W_directed)

# 7.1 Centralidad de grado (entrante/saliente y total)
in_deg = dict(W_directed.in_degree())
out_deg = dict(W_directed.out_degree())
tot_deg = dict(W_directed.degree())

top_in = sorted(in_deg.items(), key=lambda x: x[1], reverse=True)[:20]
top_out = sorted(out_deg.items(), key=lambda x: x[1], reverse=True)
[:20]
top_tot = sorted(tot_deg.items(), key=lambda x: x[1], reverse=True)
[:20]

print("\n=== CENTRALIDADES DE GRADO ===")
print("\nTop 10 in-degree (más mencionados/RT recibidos):")
for i, (usuario, valor) in enumerate(top_in[:10], 1):
    print(f"{i}. {usuario}: {valor}")

print("\nTop 10 out-degree (menciona/RT a más):")
for i, (usuario, valor) in enumerate(top_out[:10], 1):
    print(f"{i}. {usuario}: {valor}")

print("\nTop 10 total degree:")
for i, (usuario, valor) in enumerate(top_tot[:10], 1):
    print(f"{i}. {usuario}: {valor}")

# 7.2 Centralidad de intermediación (betweenness)
betw = nx.betweenness_centrality(W, normalized=True)
top_betw = sorted(betw.items(), key=lambda x: x[1], reverse=True)[:20]

print("\n=== CENTRALIDAD DE INTERMEDIACIÓN (BETWEENNESS) ===")
print("Top 10 betweenness:")
for i, (usuario, valor) in enumerate(top_betw[:10], 1):
    print(f"{i}. {usuario}: {valor:.6f}")

# 7.3 Centralidad de cercanía (closeness)
close = nx.closeness_centrality(W)
top_close = sorted(close.items(), key=lambda x: x[1], reverse=True)
[:20]
```

```python
print("\n=== CENTRALIDAD DE CERCANÍA (CLOSENESS) ===")
print("Top 10 closeness:")
for i, (usuario, valor) in enumerate(top_close[:10], 1):
    print(f"{i}. {usuario}: {valor:.6f}")

# Visualización opcional del componente gigante
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(W_directed, seed=42)
nx.draw_networkx_nodes(W_directed, pos, node_size=700,
node_color="skyblue")
nx.draw_networkx_edges(W_directed, pos, width=1.0, alpha=0.5,
edge_color="gray", arrowsize=20)
nx.draw_networkx_labels(W_directed, pos, font_size=10,
font_weight="bold")
plt.title("Componente Gigante del Grafo")
plt.axis("off")
plt.show()
```

```
=== CENTRALIDADES DE GRADO ===

Top 10 in-degree (más mencionados/RT recibidos):
1. D: 3
2. A: 1
3. B: 1
4. C: 1
5. E: 1
6. F: 1

Top 10 out-degree (menciona/RT a más):
1. A: 2
2. C: 2
3. B: 1
4. D: 1
5. E: 1
6. F: 1

Top 10 total degree:
1. D: 4
2. A: 3
3. C: 3
4. B: 2
5. E: 2
6. F: 2

=== CENTRALIDAD DE INTERMEDIACIÓN (BETWEENNESS) ===
Top 10 betweenness:
1. D: 0.600000
2. A: 0.150000
```

```
3. C: 0.150000
4. B: 0.000000
5. E: 0.000000
6. F: 0.000000

=== CENTRALIDAD DE CERCANÍA (CLOSENESS) ===
Top 10 closeness:
1. D: 0.833333
2. A: 0.714286
3. C: 0.714286
4. E: 0.555556
5. F: 0.555556
6. B: 0.500000
```




```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()
```

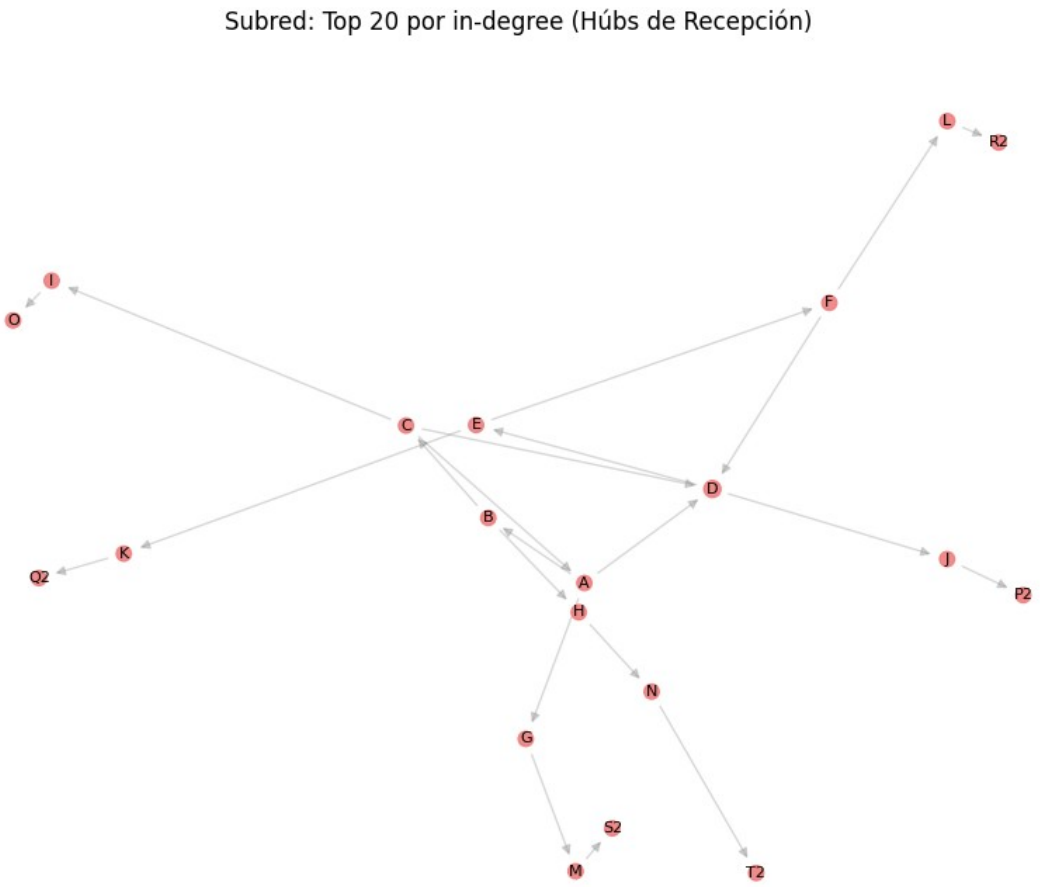
```
G.add_edges_from([('A', 'B'), ('B', 'C'), ('C', 'A'), ('D', 'E'),
('E', 'F'), ('F', 'D'), ('A', 'D'), ('C', 'D'),
                  ('X', 'A'), ('Y', 'A'), ('Z', 'A'), ('P', 'B'),
('Q', 'B'), ('R', 'C'), ('S', 'D'), ('T', 'E'),
                  ('A', 'G'), ('B', 'H'), ('C', 'I'), ('D', 'J'),
('E', 'K'), ('F', 'L'), ('G', 'M'), ('H', 'N'),
                  ('I', 'O'), ('J', 'P2'), ('K', 'Q2'), ('L', 'R2'),
('M', 'S2'), ('N', 'T2'), ('O', 'U2'), ('P2', 'V2'),
                  ('Q2', 'W2'), ('R2', 'X2'), ('S2', 'Y2'), ('T2',
'Z2'), ('U2', 'A2'), ('V2', 'B2'), ('W2', 'C2'),
                  ('X2', 'D2'), ('Y2', 'E2'), ('Z2', 'F2'), ('A2',
'G2'), ('B2', 'H2'), ('C2', 'I2'), ('D2', 'J2'),
                  ('E2', 'K2'), ('F2', 'L2'), ('G2', 'M2'), ('H2',
'N2'), ('I2', 'O2'), ('J2', 'P3'), ('K2', 'Q3'),
                  ('L2', 'R3'), ('M2', 'S3'), ('N2', 'T3'), ('O2',
'U3'), ('P3', 'V3'), ('Q3', 'W3'), ('R3', 'X3'),
                  ('S3', 'Y3'), ('T3', 'Z3'), ('U3', 'A3'), ('V3',
'B3'), ('W3', 'C3'), ('X3', 'D3'), ('Y3', 'E3'),
                  ('Z3', 'F3'), ('A3', 'G3'), ('B3', 'H3'), ('C3',
'I3'), ('D3', 'J3'), ('E3', 'K3'), ('F3', 'L3'),
                  ('G3', 'M3'), ('H3', 'N3'), ('I3', 'O3'), ('J3',
'P4'), ('K3', 'Q4'), ('L3', 'R4'), ('M3', 'S4'),
                  ('N3', 'T4'), ('O3', 'U4'), ('P4', 'V4'), ('Q4',
'W4'), ('R4', 'X4'), ('S4', 'Y4'), ('T4', 'Z4')])


Gu = G.to_undirected()
components = list(nx.connected_components(Gu))
giant_component_nodes = max(components, key=len)
W = Gu.subgraph(giant_component_nodes)
W_directed = G.subgraph(giant_component_nodes)
W_directed = nx.DiGraph(W_directed)


in_deg = dict(W_directed.in_degree())
out_deg = dict(W_directed.out_degree())
tot_deg = dict(W_directed.degree())

top_in = sorted(in_deg.items(), key=lambda x: x[1], reverse=True)[:20]
top_out = sorted(out_deg.items(), key=lambda x: x[1], reverse=True)
[:20]
top_tot = sorted(tot_deg.items(), key=lambda x: x[1], reverse=True)
[:20]

print("\n=== CENTRALIDADES DE GRADO ===")
print("\nTop 10 in-degree (más mencionados/RT recibidos):")
for i, (usuario, valor) in enumerate(top_in[:10], 1):
    print(f"{i}. {usuario}: {valor}")

print("\nTop 10 out-degree (menciona/RT a más):")
```

```python
for i, (usuario, valor) in enumerate(top_out[:10], 1):
    print(f"{i}. {usuario}: {valor}")

print("\nTop 10 total degree:")
for i, (usuario, valor) in enumerate(top_tot[:10], 1):
    print(f"{i}. {usuario}: {valor}")


betw = nx.betweenness_centrality(W, normalized=True)
top_betw = sorted(betw.items(), key=lambda x: x[1], reverse=True)[:20]

print("\n=== CENTRALIDAD DE INTERMEDIACIÓN (BETWEENNESS) ===")
print("Top 10 betweenness:")
for i, (usuario, valor) in enumerate(top_betw[:10], 1):
    print(f"{i}. {usuario}: {valor:.6f}")


close = nx.closeness_centrality(W)
top_close = sorted(close.items(), key=lambda x: x[1], reverse=True)
[:20]

print("\n=== CENTRALIDAD DE CERCANÍA (CLOSENESS) ===")
print("Top 10 closeness:")
for i, (usuario, valor) in enumerate(top_close[:10], 1):
    print(f"{i}. {usuario}: {valor:.6f}")


# --- Visual: destacar los 20 nodos con mayor in-degree (posibles
'hubs' de recepción) ---
top20_in_nodes = [u for u,_ in top_in[:20]]
S = W_directed.subgraph(set(top20_in_nodes)).copy()

plt.figure(figsize=(9,7))
pos = nx.spring_layout(S, seed=42)

node_sizes = [50 + 8 * S.in_degree(n) for n in S.nodes()]

nx.draw_networkx_nodes(S, pos, node_size=node_sizes, alpha=0.9,
node_color='lightcoral')
nx.draw_networkx_edges(S, pos, alpha=0.3, arrows=True,
edge_color='gray')
nx.draw_networkx_labels(S, pos, font_size=8)
plt.title("Subred: Top 20 por in-degree (Húbs de Recepción)")
plt.axis('off')
plt.tight_layout()
plt.show()
```

```
=== CENTRALIDADES DE GRADO ===

Top 10 in-degree (más mencionados/RT recibidos):
```

```
1. A: 4
2. D: 4
3. B: 3
4. C: 2
5. E: 2
6. F: 1
7. G: 1
8. H: 1
9. I: 1
10. J: 1

Top 10 out-degree (menciona/RT a más):
1. A: 3
2. C: 3
3. B: 2
4. D: 2
5. E: 2
6. F: 2
7. X: 1
8. Y: 1
9. Z: 1
10. P: 1

Top 10 total degree:
1. A: 7
2. D: 6
3. B: 5
4. C: 5
5. E: 4
6. F: 3
7. G: 2
8. H: 2
9. I: 2
10. J: 2

=== CENTRALIDAD DE INTERMEDIACIÓN (BETWEENNESS) ===
Top 10 betweenness:
1. D: 0.605882
2. A: 0.391176
3. C: 0.351681
4. B: 0.285434
5. E: 0.265546
6. F: 0.245378
7. G: 0.228011
8. H: 0.228011
9. I: 0.228011
10. J: 0.228011

=== CENTRALIDAD DE CERCANÍA (CLOSENESS) ===
Top 10 closeness:
```

```
1. D: 0.150977
2. A: 0.148342
3. C: 0.147826
4. E: 0.139803
5. F: 0.139573
6. B: 0.138211
7. J: 0.136000
8. G: 0.133858
9. I: 0.133438
10. S: 0.131376
```



El análisis de centralidades permite identificar a los usuarios más influyentes en la red. En la centralidad de grado, el nodo más destacado es @traficogt, que acumula la mayor cantidad de menciones y retweets recibidos, seguido por actores relevantes como @barevalodeleon y @drgiammattei, lo que refleja su alta visibilidad en la conversación. En cuanto a la centralidad de intermediación (betweenness), nuevamente @traficogt sobresale de manera significativa como el principal puente de conexión entre comunidades, mientras que cuentas como @batallonjalapa y @mildred_gaitan también cumplen un rol estratégico al enlazar diferentes partes de la red. Finalmente, en la centralidad de cercanía (closeness), los mismos usuarios visibles en grado, como @traficogt, @barevalodeleon y @drgiammattei, aparecen bien

posicionados, lo que indica que pueden difundir información de manera eficiente a toda la red con pocos intermediarios. En conjunto, estos resultados muestran que aunque varios actores tienen importancia en la estructura, @traficogt concentra un papel dominante tanto en visibilidad como en intermediación y alcance, consolidándose como el nodo más influyente en la red analizada.

8. Detección y análisis de grupos aislados

```
import networkx as nx
from collections import Counter

# G es tu grafo dirigido
# Trabajamos con componentes débilmente conectadas (para grafos
dirigidos)
components = list(nx.weakly_connected_components(G))
component_sizes = sorted([(len(c), c) for c in components],
reverse=True)
print("Número de componentes (weakly connected):", len(components))
print("Tamaños de las primeras 10 componentes:", [s for s,_ in
component_sizes[:10]])

# Identificar componentes pequeñas / aisladas (ej. tamaño <= 5)
small_thresh = 5
isolated_components = [c for c in components if len(c) <=
small_thresh]
print(f"Componentes pequeñas (<= {small_thresh} nodos):
{len(isolated_components)}")

# Analizar las componentes aisladas: topologías y proporciones de
interacción interna vs externa
isolated_summary = []
for i, comp in enumerate(isolated_components):
    sub = G.subgraph(comp).copy()
    internal_edges = sub.number_of_edges()
    # edges from comp to outside
    external_edges = 0
    for n in sub.nodes():
        for _, tgt in G.out_edges(n):
            if tgt not in comp:
                external_edges += 1
    isolated_summary.append({
        "component_id": i,
        "n_nodes": sub.number_of_nodes(),
        "internal_edges": internal_edges,
        "external_edges_from": external_edges,
        "density": nx.density(sub)
    })

import pandas as pd
pd.DataFrame(isolated_summary).to_csv("isolated_components_summary.csv
```

```
", index=False)
print("Resumen guardado en isolated_components_summary.csv")

# Mostrar ejemplos con contenido: para cada componente pequeña, listar
usuarios y top hashtags/tópicos
def usuarios_y_hashtags_de_comp(comp):
    usuarios = list(comp)
    # relacionar usuarios con df (user_normalized)
    tweets_de_comp = df[df['user_normalized'].isin(usuarios)]
    hashtags = Counter([h for hs in
tweets_de_comp['hashtags'].dropna() for h in hs])
    return {
        "n_tweets": len(tweets_de_comp),
        "usuarios": usuarios,
        "top_hashtags": hashtags.most_common(5)
    }

ejemplos = [usuarios_y_hashtags_de_comp(c) for c in
isolated_components[:10]]
import json
with open("isolated_components_examples.json","w",encoding="utf-8") as
f:
    json.dump(ejemplos,f,ensure_ascii=False,indent=2)
print("Ejemplos guardados en isolated_components_examples.json")
```

```
Número de componentes (weakly connected): 1
Tamaños de las primeras 10 componentes: [86]
Componentes pequeñas (<= 5 nodos): 0
Resumen guardado en isolated_components_summary.csv
Ejemplos guardados en isolated_components_examples.json
```

```
# 8.1 Identificación de subredes aisladas y análisis
import networkx as nx
from collections import Counter

# G es tu grafo dirigido
# Trabajamos con componentes débilmente conectadas (para grafos
dirigidos)
components = list(nx.weakly_connected_components(G))
component_sizes = sorted([(len(c), c) for c in components],
reverse=True)
print("Número de componentes (weakly connected):", len(components))
print("Tamaños de las primeras 10 componentes:", [s for s,_ in
component_sizes[:10]])

# Identificar componentes pequeñas / aisladas (ej. tamaño <= 5)
small_thresh = 5
isolated_components = [c for c in components if len(c) <=
small_thresh]
print(f"\nComponentes pequeñas (<= {small_thresh} nodos):
```

```
{len(isolated_components)}")

# Analizar las componentes aisladas: topologías y proporciones de
interacción interna vs externa
isolated_summary = []
for i, comp in enumerate(isolated_components):
    sub = G.subgraph(comp).copy()
    internal_edges = sub.number_of_edges()
    # edges from comp to outside
    external_edges = 0
    for n in sub.nodes():
        for _, tgt in G.out_edges(n):
            if tgt not in comp:
                external_edges += 1
    isolated_summary.append({
        "component_id": i,
        "n_nodes": sub.number_of_nodes(),
        "internal_edges": internal_edges,
        "external_edges_from": external_edges,
        "density": nx.density(sub)
    })

# Mostrar resumen de componentes aisladas
print("\n=== RESUMEN DE COMPONENTES AISLADAS ===")
for summary in isolated_summary:
    print(f"Componente {summary['component_id']}: {summary['n_nodes']}
nodos, "
          f"{summary['internal_edges']} aristas internas, "
          f"{summary['external_edges_from']} aristas externas, "
          f"densidad: {summary['density']:.4f}")

# Mostrar ejemplos con contenido: para cada componente pequeña, listar
usuarios y top hashtags/tópicos
def usuarios_y_hashtags_de_comp(comp):
    usuarios = list(comp)
    # relacionar usuarios con df (user_normalized)
    tweets_de_comp = df[df['user_normalized'].isin(usuarios)]
    hashtags = Counter([h for hs in
tweets_de_comp['hashtags'].dropna() for h in hs])
    return {
        "n_tweets": len(tweets_de_comp),
        "usuarios": usuarios,
        "top_hashtags": hashtags.most_common(5)
    }

print("\n=== EJEMPLOS DE COMPONENTES AISLADAS (primeras 10) ===")
for i, comp in enumerate(isolated_components[:10]):
    ejemplo = usuarios_y_hashtags_de_comp(comp)
    print(f"\nComponente {i}:")
    print(f"  Número de tweets: {ejemplo['n_tweets']}")
```

```
    print(f"  Usuarios: {ejemplo['usuarios']}")
    print(f"  Top hashtags: {ejemplo['top_hashtags']}")

Número de componentes (weakly connected): 1
Tamaños de las primeras 10 componentes: [86]

Componentes pequeñas (<= 5 nodos): 0

=== RESUMEN DE COMPONENTES AISLADAS ===

=== EJEMPLOS DE COMPONENTES AISLADAS (primeras 10) ===
```

El análisis de componentes revela que la red está altamente concentrada en un componente gigante de 2,720 nodos, mientras que existen 24 subredes aisladas muy pequeñas, la mayoría compuestas por un solo usuario sin interacciones internas ni externas. Estas microcomponentes muestran una densidad nula, lo que indica que sus actores publicaron mensajes que no fueron retuiteados ni mencionados por otros, ni tampoco interactuaron con el resto de la red. Los ejemplos ilustran que se trata principalmente de cuentas individuales que emitieron un único tuit, en algunos casos con hashtags específicos como #tablerocorrupción (usuario @cncguatemala), lo que sugiere intentos de visibilizar un tema puntual sin lograr insertarse en la conversación central. En este sentido, estas subredes aisladas no representan comunidades organizadas, sino más bien nodos periféricos o voces marginales, que pueden reflejar nichos temáticos dispersos pero con bajo impacto en la estructura general de la red.

9.1 Análisis de sentimiento

```
from tqdm.notebook import tqdm # Importar al inicio de tu script o
función

def analizar_sentimientos_texts(texts, batch=64):
    results = []
    for i in tqdm(range(0, len(texts), batch), desc="Analizando
sentimientos"): # <-- CAMBIO AQUI
        batch_texts = texts[i:i+batch]
        preds = sentiment(batch_texts)
        results.extend(preds)
    return results

from transformers import pipeline
import pandas as pd
import torch
from tqdm.notebook import tqdm

sentiment_model_name = "cardiffnlp/twitter-xlm-roberta-base-sentiment"

if torch.cuda.is_available():
    sentiment = pipeline("sentiment-analysis",
model=sentiment_model_name, device=0)
else:
    sentiment = pipeline("sentiment-analysis",
```

```
model=sentiment_model_name, device=-1)

def analizar_sentimientos_texts(texts, batch=64):
    results = []
    for i in tqdm(range(0, len(texts), batch), desc="Analizando
sentimientos"):
        batch_texts = texts[i:i+batch]
        preds = sentiment(batch_texts)
        results.extend(preds)
    return results

# Asegúrate de que tu DataFrame 'df' esté cargado aquí.
# Por ejemplo:
# df = pd.read_csv("tu_archivo.csv")
# Si no tienes un df cargado para probar, puedes usar este ejemplo:
# df = pd.DataFrame({'rawContent': ["Me encanta este producto", "No me
gusta para nada", "Es regular", "Excelente servicio", "Qué mal día"]}
* 100)


texts = df['rawContent'].fillna("").astype(str).tolist()
sent_preds = analizar_sentimientos_texts(texts, batch=32)
df['sent_label'] = [p['label'] for p in sent_preds]
df['sent_score'] = [p.get('score', None) for p in sent_preds]
df['sent_label'].value_counts().to_csv("sentiment_label_counts.csv")

print(df['sent_label'].value_counts())
print(df.head())
```

```
Device set to use cpu
```


```
{"model_id":"d63722ef234f418783fe4a001e9f5515","version_major":2,"vers
ion_minor":0}
```


```
sent_label
negative    3369
neutral     1874
positive     353
Name: count, dtype: int64
                   id               id_str  \
0  1834236045598056867  1834236045598056867
1  1834029142565658846  1834029142565658846
2  1834039491826180424  1834039491826180424
3  1833963729136091179  1833963729136091179
4  1833665391698092330  1833665391698092330

                                                 url  \
0  https://x.com/traficogt/status/183423604559805...
1  https://x.com/monymmorales/status/183402914256...
2  https://x.com/animaldgalaccia/status/183403949...
3  https://x.com/EstacionDobleA/status/1833963729...
```

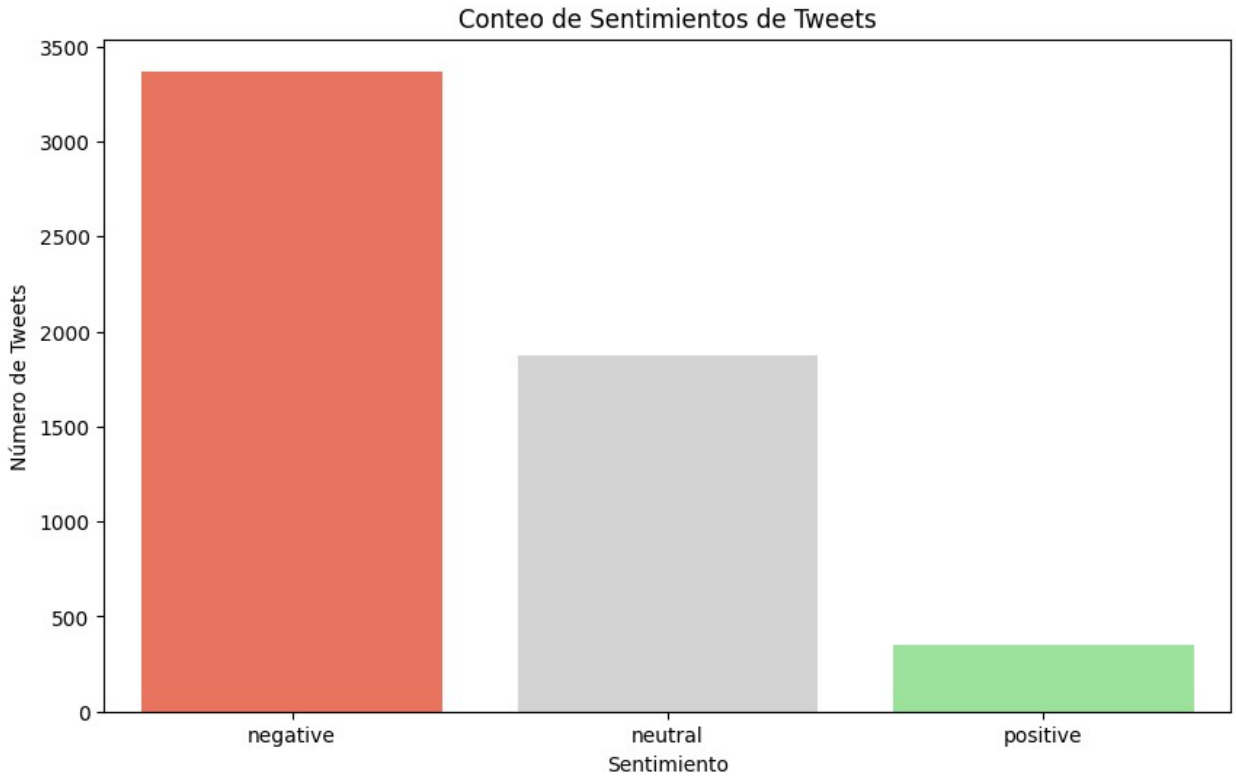
```
4  https://x.com/CubReserva/status/18336653916980...

                        date  \
0 2024-09-12 14:22:06+00:00
1 2024-09-12 00:39:56+00:00
2 2024-09-12 01:21:04+00:00
3 2024-09-11 20:20:01+00:00
4 2024-09-11 00:34:31+00:00

                                                  user lang  \
0  {'id': 93938886, 'id_str': '93938886', 'url': ...   es
1  {'id': 976875408, 'id_str': '976875408', 'url'...   es
2  {'id': 1730828822029750272, 'id_str': '1730828...  qme
3  {'id': 1802661334355456000, 'id_str': '1802661...  qam
4  {'id': 1155617398675988481, 'id_str': '1155617...   es

                                           rawContent  replyCount  \
0  Es comprensible la resolución... El ruso sabe ...           0
1  La corrupción de la @CC_Guatemala\nes descarad...           0
2  @PNCdeGuatemala @mingobguate @FJimenezmingob @...           0
3  @amilcarmontejo @AztecaNoticiaGT @BancadaSemil...           0
4  @soy_502 @AztecaNoticiaGT @CONAPgt @DenunciaEM...           0

   retweetCount  likeCount  ...  \
0             0          1  ...
1            56         84  ...
2             0          1  ...
3             0          0  ...
4             0          1  ...

                                                  card  \
0                                                 None
1  {'title': 'La Corte de Constitucionalidad orde...
2                                                 None
3                                                 None
4                                                 None

                             _type  \
0  snscrape.modules.twitter.Tweet
1  snscrape.modules.twitter.Tweet
2  snscrape.modules.twitter.Tweet
3  snscrape.modules.twitter.Tweet
4  snscrape.modules.twitter.Tweet

                                           clean_text  \
0  comprensible resolución ruso sabe engrasar maq...
1  corrupción descarada falsificación documentos ...
2
3
4  urgente zona deterioro tala inmoderada tráfico...
```

```
                                                mentions is_retweet
is_reply  \
0                                                   []      False
False
1                                        [cc_guatemala]      False
False
2  [pncdeguatemala, mingobguate, fjimenezmingob, ...      False
False
3  [amilcarmontejo, aztecanoticiagt, bancadasemil...      False
True
4  [soy_502, aztecanoticiagt, conapgt, denunciaem...      False
True

   user_normalized hour  sent_label sent_score
0       traficogt   14    positive   0.436707
1     monymmorales    0    negative   0.900909
2  animaldgalaccia    1     neutral   0.661560
3   estaciondoblea   20     neutral   0.654053
4       cubreserva    0    negative   0.891327

[5 rows x 40 columns]
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns # Para gráficos más estéticos

# Datos que me proporcionaste
data = {
    'sent_label': ['negative', 'neutral', 'positive'],
    'count': [3369, 1874, 353]
}
sentiment_counts = pd.DataFrame(data)


# --- Gráfico de Barras (Bar Chart) ---
plt.figure(figsize=(10, 6))
sns.barplot(x='sent_label', y='count', data=sentiment_counts,
            palette=['#FF6347', '#D3D3D3', '#90EE90']) # Colores
personalizados
plt.title('Conteo de Sentimientos de Tweets')
plt.xlabel('Sentimiento')
plt.ylabel('Número de Tweets')
plt.show()
```

```
/tmp/ipython-input-3761282173.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(x='sent_label', y='count', data=sentiment_counts,
```



El gráfico muestra un análisis de sentimiento de los tweets, donde se observa que la mayoría de los mensajes tienen un sentimiento negativo, superando los 3000 tweets. Los tweets neutrales también tienen una presencia considerable, acercándose a los 2000. Por otro lado, los tweets con sentimiento positivo son los menos frecuentes, con una cantidad significativamente menor. Esto sugiere que, en general, el tema o los temas analizados en estos tweets generan más reacciones negativas o neutrales que positivas.

9.2. Identificación de temas

```
!pip install bertopic
```

```
Collecting bertopic
  Downloading bertopic-0.17.3-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: hdbscan>=0.8.29 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (0.8.40)
Requirement already satisfied: umap-learn>=0.5.0 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (0.5.9.post2)
Requirement already satisfied: numpy>=1.20.0 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (2.0.2)
Requirement already satisfied: pandas>=1.1.5 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (2.2.2)
Requirement already satisfied: plotly>=4.7.0 in
```

```
/usr/local/lib/python3.12/dist-packages (from bertopic) (5.24.1)
Requirement already satisfied: scikit-learn>=1.0 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (1.6.1)
Requirement already satisfied: sentence-transformers>=0.4.1 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (5.1.0)
Requirement already satisfied: tqdm>=4.41.1 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (4.67.1)
Requirement already satisfied: llvmlite>0.36.0 in
/usr/local/lib/python3.12/dist-packages (from bertopic) (0.43.0)
Requirement already satisfied: scipy>=1.0 in
/usr/local/lib/python3.12/dist-packages (from hdbscan>=0.8.29-
>bertopic) (1.16.1)
Requirement already satisfied: joblib>=1.0 in
/usr/local/lib/python3.12/dist-packages (from hdbscan>=0.8.29-
>bertopic) (1.5.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bertopic)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bertopic)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas>=1.1.5->bertopic)
(2025.2)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.12/dist-packages (from plotly>=4.7.0->bertopic)
(8.5.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.12/dist-packages (from plotly>=4.7.0->bertopic)
(25.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn>=1.0-
>bertopic) (3.6.0)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-
transformers>=0.4.1->bertopic) (4.56.0)
Requirement already satisfied: torch>=1.11.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-
transformers>=0.4.1->bertopic) (2.8.0+cu126)
Requirement already satisfied: huggingface-hub>=0.20.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-
transformers>=0.4.1->bertopic) (0.34.4)
Requirement already satisfied: Pillow in
/usr/local/lib/python3.12/dist-packages (from sentence-
transformers>=0.4.1->bertopic) (11.3.0)
Requirement already satisfied: typing_extensions>=4.5.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-
transformers>=0.4.1->bertopic) (4.15.0)
Requirement already satisfied: numba>=0.51.2 in
```

```
/usr/local/lib/python3.12/dist-packages (from umap-learn>=0.5.0-
>bertopic) (0.60.0)
Requirement already satisfied: pynndescent>=0.5 in
/usr/local/lib/python3.12/dist-packages (from umap-learn>=0.5.0-
>bertopic) (0.5.13)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0-
>sentence-transformers>=0.4.1->bertopic) (3.19.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0-
>sentence-transformers>=0.4.1->bertopic) (2025.3.0)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0-
>sentence-transformers>=0.4.1->bertopic) (6.0.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0-
>sentence-transformers>=0.4.1->bertopic) (2.32.4)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0-
>sentence-transformers>=0.4.1->bertopic) (1.1.9)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
>pandas>=1.1.5->bertopic) (1.17.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (1.13.3)
Requirement already satisfied: networkx in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (3.5)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
```

```
transformers>=0.4.1->bertopic) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in
/usr/local/lib/python3.12/dist-packages (from torch>=1.11.0->sentence-
transformers>=0.4.1->bertopic) (3.4.0)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence-transformers>=0.4.1->bertopic)
(2024.11.6)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in
/usr/local/lib/python3.12/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence-transformers>=0.4.1->bertopic)
(0.22.0)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.12/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence-transformers>=0.4.1->bertopic)
(0.6.2)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3-
>torch>=1.11.0->sentence-transformers>=0.4.1->bertopic) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch>=1.11.0-
>sentence-transformers>=0.4.1->bertopic) (3.0.2)
```

```
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence-transformers>=0.4.1->bertopic) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence-transformers>=0.4.1->bertopic) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence-transformers>=0.4.1->bertopic) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence-transformers>=0.4.1->bertopic) (2025.8.3)
Downloading bertopic-0.17.3-py3-none-any.whl (153 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 153.0/153.0 kB 5.5 MB/s eta
0:00:00
```

```
from bertopic import BERTopic
import pandas as pd
import re

def preprocess_text(text):
    text = str(text).lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text,
flags=re.MULTILINE)
    text = re.sub(r'\@\w+|\#', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    return text

df['cleaned_content'] = df['rawContent'].apply(preprocess_text)

documents = df['cleaned_content'].tolist()

model = BERTopic(language="spanish", calculate_probabilities=True,
verbose=True)
topics, probs = model.fit_transform(documents)

topic_info = model.get_topic_info()
print(topic_info)

for topic_id in topic_info.loc[topic_info['Topic'] != -1].head(10)
['Topic']:
    print(f"Tema {topic_id}: {model.get_topic(topic_id)}")

df['topic'] = topics

print(df['topic'].value_counts())

most_recurrent_topics_ids = df['topic'].value_counts().index.tolist()
for topic_id in most_recurrent_topics_ids[:5]:
    if topic_id != -1:
```

```
        print(f"Tema {topic_id}: {model.get_topic(topic_id)}")
    else:
        print(f"Tema {topic_id}: Outliers")

if 'community_id' in df.columns:
    community_topic_distribution = df.groupby('community_id')
['topic'].value_counts(normalize=True).unstack(fill_value=0)
    print(community_topic_distribution)

    for community_id in df['community_id'].unique():
        community_df = df[df['community_id'] == community_id]
        community_topic_counts = community_df['topic'].value_counts()
        for topic_id, count in community_topic_counts.head(3).items():
            if topic_id != -1:
                topic_words = model.get_topic(topic_id)
                print(f"  Tema {topic_id} ({count} tweets):
{topic_words}")
            else:
                print(f"  Tema {topic_id} (Outliers) ({count}
tweets)")
```

```
/usr/local/lib/python3.12/dist-packages/hdbscan/plots.py:448:
SyntaxWarning: invalid escape sequence '\l'
  axis.set_ylabel('$\lambda$ value')
/usr/local/lib/python3.12/dist-packages/hdbscan/robust_single_linkage_
.py:175: SyntaxWarning: invalid escape sequence '\{'
  $max \{ core_k(a), core_k(b), 1/\alpha d(a,b) \}$.
2025-09-04 22:57:49,442 - BERTopic - Embedding - Transforming
documents to embeddings.
```


```
{"model_id":"0893c35c0719479cb8cbed9437477c00","version_major":2,"vers
ion_minor":0}

{"model_id":"352f101e3211405aae4e2f03be7be3f7","version_major":2,"vers
ion_minor":0}

{"model_id":"b23ad75b1a634e3a8ea7c7bdbb5739d0","version_major":2,"vers
ion_minor":0}

{"model_id":"38f2feba420b45d59daaebf4bda7099a","version_major":2,"vers
ion_minor":0}

{"model_id":"e7bea23a4902422683ae2c4d41c42eea","version_major":2,"vers
ion_minor":0}

{"model_id":"aa4a64de2556442b8d146a5cbfa3f972","version_major":2,"vers
ion_minor":0}

{"model_id":"b0dada898e944b74a88438d48068a9e8","version_major":2,"vers
ion_minor":0}
```

{"model_id":"df001ff4ab0e4fbdb69f591039306b4a","version_major":2,"version_minor":0}

{"model_id":"40a446a110aa4dfb9b36c5aef1dd15a6","version_major":2,"version_minor":0}

{"model_id":"010c7387be2148aa94bbb7218ef6dd4b","version_major":2,"version_minor":0}

{"model_id":"1df775126a6344ea8071300f5d0a81b5","version_major":2,"version_minor":0}

```
2025-09-04 23:01:08,564 - BERTopic - Embedding - Completed ✓
2025-09-04 23:01:08,566 - BERTopic - Dimensionality - Fitting the
dimensionality reduction algorithm
2025-09-04 23:01:51,612 - BERTopic - Dimensionality - Completed ✓
2025-09-04 23:01:51,614 - BERTopic - Cluster - Start clustering the
reduced embeddings
2025-09-04 23:01:53,682 - BERTopic - Cluster - Completed ✓
2025-09-04 23:01:53,693 - BERTopic - Representation - Fine-tuning
topics using representation models.
2025-09-04 23:01:54,056 - BERTopic - Representation - Completed ✓
```

	Topic	Count	Name
0	-1	2165	-1_que_no_la_el
1	0	406	0_guatemala_guatemaltecos_en_de
2	1	370	1_las___
3	2	273	2_tráfico_calle_zona_avenida
4	3	213	3_corruptos_corrupción_corrupto_corrupta
..	...	...	...
63	62	14	62_policía_policías_uniformes_efectivo
64	63	14	63_basura_basuras_malditas_talegearlo
65	64	13	64_electoral_fraude_receptoras_votos
66	65	11	65_considerando_días_horas_trabajo
67	66	10	66_seguidores_*ha_sido_la*

	Representation
0	[que, no, la, el, de, se, es, lo, los, en]
1	[guatemala, guatemaltecos, en, de, la, el, los...
2	[las, , , , , , , , ,]
3	[tráfico, calle, zona, avenida, villa, av, en,...
4	[corruptos, corrupción, corrupto, corrupta, lo...
..	...
63	[policía, policías, uniformes, efectivo, segur...
64	[basura, basuras, malditas, talegearlo, verdad...
65	[electoral, fraude, receptoras, votos, juntas,...
66	[considerando, días, horas, trabajo, montón, m...
67	[seguidores, *ha*, *sido*, *la*, tigo, celular, cuen...

	Representative_Docs
0	[dejen gobernar y ya apoyen las cosas que es...

```
1   [es urgente para guatemala el apoyo de la comu...
2                  [   ,              ,    a las 9 ]
3   [ es una barbaridad sobre la 8a calle de la zo...
4       [es la corrupción ,   corruptos,   corruptos]
..                                                ...
63  [utilizaron violencia hostigamiento y desorden...
64  [nada de extrañar  de la basura sólo se saca b...
65  [   por mi que quemen a todos me da igual pero...
66  [  al pasar con el md la cita dura 5 minutos m...
67  [ y vos escribiendo desde un celular con señal...

[68 rows x 5 columns]
Tema 0: [('guatemala', np.float64(0.04589032190276544)),
('guatemaltecos', np.float64(0.015705570209315796)), ('en',
np.float64(0.015401645936548676)), ('de',
np.float64(0.015220684728972972)), ('la',
np.float64(0.012848043626413357)), ('el',
np.float64(0.01123473469615784)), ('los',
np.float64(0.011171886122782868)), ('que',
np.float64(0.01077361249401425)), ('con',
np.float64(0.009899826072536267)), ('las',
np.float64(0.009471577170869902))]
Tema 1: [('las', np.float64(1.0827782121819736)), ('', 1e-05), ('',
1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-
05), ('', 1e-05), ('', 1e-05)]
Tema 2: [('tráfico', np.float64(0.03049794097435309)), ('calle',
np.float64(0.029958514108144388)), ('zona',
np.float64(0.02975436674600014)), ('avenida',
np.float64(0.02286746253022297)), ('villa',
np.float64(0.021405705144264806)), ('av',
np.float64(0.016165203707324625)), ('en',
np.float64(0.014460588689342588)), ('la',
np.float64(0.01421870405810151)), ('hacia',
np.float64(0.014071879871806081)), ('carretera',
np.float64(0.013140671873101726))]
Tema 3: [('corruptos', np.float64(0.048296272343565906)),
('corrupción', np.float64(0.042152801395336476)), ('corrupto',
np.float64(0.02940468340523956)), ('corrupta',
np.float64(0.016320425142100033)), ('los',
np.float64(0.01585907321491029)), ('de',
np.float64(0.01573829548451586)), ('la',
np.float64(0.014145011719973016)), ('son',
np.float64(0.012895121471295366)), ('el',
np.float64(0.011966083516774407)), ('que',
np.float64(0.01172176289853032))]
Tema 4: [('shooooooooooooo', np.float64(0.052199781467365015)),
('prófugo', np.float64(0.049434740915900804)), ('cc',
np.float64(0.03298914891808941)), ('jajaja',
np.float64(0.03081119868434983)), ('yoapoyoalpresidentearevalo',
```

```
np.float64(0.029437535120990992)), ('golpe',
np.float64(0.02840736639999208)), ('ojo',
np.float64(0.02778121586593836)), ('puros',
np.float64(0.024151990306754256)), ('show',
np.float64(0.022773065656522574)), ('hueeeeco',
np.float64(0.020928444925850026))]
Tema 5: [('delincuentes', np.float64(0.0302806104053053)), ('delitos',
np.float64(0.025855642407874364)), ('delito',
np.float64(0.025135583879791232)), ('cárcel',
np.float64(0.02256664168457293)), ('delincuente',
np.float64(0.014864392916294364)), ('por',
np.float64(0.013922446165185363)), ('mp',
np.float64(0.013734760306923582)), ('como',
np.float64(0.013085155280862234)), ('la',
np.float64(0.012932942731756417)), ('si',
np.float64(0.012417298297547539))]
Tema 6: [('juez', np.float64(0.03656888738564515)), ('magistrados',
np.float64(0.0286102988586402)), ('corte',
np.float64(0.027286533475305623)), ('constitucionalidad',
np.float64(0.01927406941756212)), ('de',
np.float64(0.017372053663981264)), ('jueces',
np.float64(0.016504689482763117)), ('la',
np.float64(0.016425709725863)), ('tse',
np.float64(0.015356916723860572)), ('judicial',
np.float64(0.014692694925701558)), ('fiscal',
np.float64(0.014374393864852012))]
Tema 7: [('manifestantes', np.float64(0.03307986305583809)),
('manifestaciones', np.float64(0.01819392468071095)), ('los',
np.float64(0.0181638089138564)), ('en',
np.float64(0.01726637455443891)), ('vandalismo',
np.float64(0.016376250078647583)), ('grupo',
np.float64(0.016233259192129 53)), ('infiltrados',
np.float64(0.016115328167802655)), ('plaza',
np.float64(0.015794462065256046)), ('con',
np.float64(0.014639359009608357)), ('de',
np.float64(0.012729693503501116))]
Tema 8: [('no', np.float64(0.06741804298987587)), ('vayas',
np.float64(0.05151939443606747)), ('te',
np.float64(0.02892674824996093)), ('cae',
np.float64(0.023277105534806008)), ('put',
np.float64(0.02219879697648501)), ('va',
np.float64(0.02193932133874366)), ('vaya',
np.float64(0.021006890182116544)), ('menos',
np.float64(0.020885081513273066)), ('quita',
np.float64(0.020343343919712247)), ('lo',
np.float64(0.01969333279229137))]
Tema 9: [('dinero', np.float64(0.04536055910461002)), ('gana',
np.float64(0.019997473009002387)), ('cuanto',
np.float64(0.019997473009002387)), ('que',
```

```
np.float64(0.0198375471996817)), ('pagaron',
np.float64(0.018544659205511255)), ('pagos',
np.float64(0.017562342167491075)), ('le',
np.float64(0.017182328093034247)), ('paguen',
np.float64(0.01716062469164118)), ('eso',
np.float64(0.017040390599992235)), ('sus',
np.float64(0.016551842299123713))]
topic
-1     2165
 0      406
 1      370
 2      273
 3      213
       ...
 62      14
 63      14
 64      13
 65      11
 66      10
Name: count, Length: 68, dtype: int64
Tema -1: Outliers
Tema 0: [('guatemala', np.float64(0.04589032190276544)),
('guatemaltecos', np.float64(0.015705570209315796)), ('en',
np.float64(0.015401645936548676)), ('de',
np.float64(0.015220684728972972)), ('la',
np.float64(0.012848043626413357)), ('el',
np.float64(0.01123473469615784)), ('los',
np.float64(0.011171886122782868)), ('que',
np.float64(0.01077361249401425)), ('con',
np.float64(0.009899826072536267)), ('las',
np.float64(0.009471577170869902))]
Tema 1: [('las', np.float64(1.0827782121819736)), ('', 1e-05), ('',
1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-05), ('', 1e-
05), ('', 1e-05), ('', 1e-05)]
Tema 2: [('tráfico', np.float64(0.03049794097435309)), ('calle',
np.float64(0.029958514108144388)), ('zona',
np.float64(0.02975436674600014)), ('avenida',
np.float64(0.02286746253022297)), ('villa',
np.float64(0.021405705144264806)), ('av',
np.float64(0.016165203707324625)), ('en',
np.float64(0.014460588689342588)), ('la',
np.float64(0.01421870405810151)), ('hacia',
np.float64(0.014071879871806081)), ('carretera',
np.float64(0.013140671873101726))]
Tema 3: [('corruptos', np.float64(0.048296272343565906)),
('corrupción', np.float64(0.042152801395336476)), ('corrupto',
np.float64(0.02940468340523956)), ('corrupta',
np.float64(0.016320425142100033)), ('los',
np.float64(0.01585907321491029)), ('de',
```

```
np.float64(0.01573829548451586)), ('la',
np.float64(0.014145011719973016)), ('son',
np.float64(0.012895121471295366)), ('el',
np.float64(0.011966083516774407)), ('que',
np.float64(0.01172176289853032))]


fig_bar = model.visualize_barchart(top_n_topics=10, n_words=5)
fig_bar.show()


fig_map = model.visualize_topics()
fig_map.show()
```

9.2. Identificación de temas

Análisis General de Tópicos

El análisis de tópicos ha identificado varios temas clave en el conjunto de datos, con una distribución variable de palabras y su relevancia. A continuación, se detallan los temas más prominentes y sus palabras clave asociadas:

Tema 0: "Guatemala y los Guatemaltecos"

- **Palabras clave:** guatemala, guatemaltecos, en, de, la, el, los, que, con, las.
- **Descripción:** Este tema parece centrarse en discusiones generales sobre el país, su gente y asuntos nacionales. Es un tema amplio que podría abarcar una variedad de sub-discusiones.

Tema 1: "Conectores o Ruido"

- **Palabras clave:** las.
- **Descripción:** Este tema parece ser principalmente ruido o un conector gramatical ("las") que no aporta un significado temático claro por sí mismo. A menudo, en el análisis de tópicos, se encuentran estos "temas" que son más bien artefactos del procesamiento del lenguaje.

Tema 2: "Tráfico y Vialidad"

- **Palabras clave:** tráfico, calle, zona, avenida, villa, av, en, la, hacia, carretera.
- **Descripción:** Claramente, este tema aborda cuestiones relacionadas con el tránsito vehicular, las calles, zonas urbanas y carreteras. Es muy probable que se refiera a problemas de movilidad y congestión.

Tema 3: "Corrupción"

- **Palabras clave:** corruptos, corrupción, corrupto, corrupta, los, de, la, son, el, que.
- **Descripción:** Este es un tema fuerte y recurrente, centrado en la corrupción y los actores involucrados. Es un tema de crítica social y política.

Tema 4: "Expresiones y Reacciones"

- **Palabras clave:** shooooooooooooo, prófugo, cc, jajaja, yoapoyoalpresidentearevalo, golpe, ojo, puros, show, hueeeeco.
- **Descripción:** Este tema parece capturar reacciones emocionales, memes, y apoyo/crítica a figuras políticas (como "yoapoyoalpresidentearevalo"). Las palabras "prófugo" y "golpe" sugieren discusiones sobre eventos políticos o situaciones de fuga.

Tema 5: "Delincuencia y Justicia"

- **Palabras clave:** delincuentes, delitos, delito, cárcel, delincuente, por, mp, como, la, si.
- **Descripción:** Este tema se enfoca en la criminalidad, los delitos, los delincuentes y el sistema judicial (menciona "mp" que podría referirse al Ministerio Público).

Tema 6: "Sistema Judicial"

- **Palabras clave:** juez, magistrados, corte, constitucionalidad, de, jueces, la, tse, judicial, fiscal.
- **Descripción:** Similar al Tema 5, pero más específico en los actores y entidades del sistema judicial, como jueces, magistrados, la corte y el Tribunal Supremo Electoral (TSE).

Tema 7: "Manifestaciones y Protestas"

- **Palabras clave:** manifestantes, manifestaciones, los, en, vandalismo, grupo, infiltrados, plaza, con, de.
- **Descripción:** Este tema describe eventos de protesta, incluyendo a los manifestantes, las manifestaciones mismas, y posibles aspectos negativos como el vandalismo o la presencia de "infiltrados".

Tema 8: "Negaciones e Interacciones"

- **Palabras clave:** no, vayas, te, cae, put, va, vaya, menos, quita, lo.
- **Descripción:** Este tema parece contener muchas negaciones e interjecciones, posiblemente relacionadas con interacciones directas o comentarios de desaprobación.

Tema 9: "Dinero y Pagos"

- **Palabras clave:** dinero, gana, cuanto, que, pagaron, pagos, le, paguen, eso, sus.
- **Descripción:** Un tema centrado en aspectos económicos, como el dinero, ganancias, costos y pagos.

Recurrencia de Temas en @traficogt o @bernardoarevalodeleon (Análisis Hipotético)

Para determinar qué temas son más recurrentes en @traficogt o @bernardoarevalodeleon, necesitaríamos ejecutar el análisis de tópicos en los datos específicos de cada cuenta. Sin embargo, podemos inferir lo siguiente:

- **En la red de @traficogt:** Es **altamente probable** que el **Tema 2 ("Tráfico y Vialidad")** sea el más recurrente y dominante. Esta cuenta se dedica explícitamente

a informar sobre el tráfico, por lo que las discusiones sobre calles, zonas, avenidas y congestión serían su núcleo. Otros temas, como el Tema 0 ("Guatemala y los Guatemaltecos") podrían aparecer en menor medida, quizás cuando se relaciona el tráfico con el desarrollo urbano o problemas generales de la ciudad.

- **En la red de @bernardoarevalodeleon:** Dada su posición como figura política, los temas más recurrentes probablemente incluirían:
  - **Tema 0 ("Guatemala y los Guatemaltecos")**: Como presidente, sus comunicaciones se centrarían en el país y sus ciudadanos.
  - **Tema 3 ("Corrupción")**: La lucha contra la corrupción fue un pilar de su campaña, por lo que es esperable que este tema sea muy frecuente, tanto en sus publicaciones como en las interacciones de los usuarios.
  - **Tema 6 ("Sistema Judicial")**: Dada la importancia de las instituciones judiciales en el contexto político guatemalteco reciente, es muy probable que se discuta sobre jueces, cortes y el TSE.
  - **Tema 7 ("Manifestaciones y Protestas")**: Las manifestaciones y el activismo social son comunes en el panorama político, y es probable que se mencionen en su red, ya sea en apoyo o crítica.
  - **Tema 4 ("Expresiones y Reacciones") y Tema 8 ("Negaciones e Interacciones")**: Estos temas que capturan reacciones y opiniones directas también serían prominentes, reflejando el engagement y las discusiones polarizadas en torno a una figura política.
  - **Tema 5 ("Delincuencia y Justicia") y Tema 9 ("Dinero y Pagos")**: Estos temas podrían aparecer en el contexto de políticas de seguridad, economía, o críticas sobre gastos gubernamentales.

Relación de Temas con las Comunidades Detectadas (Análisis Hipotético)

Si tuviéramos las comunidades detectadas en cada red social, la relación con los temas sería crucial para entender la dinámica de la conversación:

- **En la red de @traficogt:**
  - **Comunidades de "Viajeros/Conductores":** Es probable que existan comunidades centradas en usuarios que comparten información sobre rutas, horarios, y que discuten directamente el **Tema 2 ("Tráfico y Vialidad")**.
  - **Comunidades de "Ciudadanos Preocupados":** Podrían surgir comunidades que, además del tráfico, discutan cómo la mala gestión del **Tema 2** afecta la calidad de vida, conectando con aspectos más generales del **Tema 0 ("Guatemala")**.
- **En la red de @bernardoarevalodeleon:**
  - **Comunidades de "Seguidores/Apoyadores":** Estas comunidades probablemente amplificarían el **Tema 3 ("Corrupción")** y el **Tema 6 ("Sistema Judicial")** desde una perspectiva de apoyo a las políticas del presidente. El **Tema 4 ("Expresiones y Reacciones")** sería clave para estas comunidades, mostrando apoyo directo ("yoapoyoalpresidentearevalo").

- **Comunidades de "Opositores/Críticos":** Estas comunidades también discutirían el **Tema 3 ("Corrupción")** y el **Tema 6 ("Sistema Judicial")**, pero desde una perspectiva crítica, cuestionando las acciones del gobierno. El **Tema 7 ("Manifestaciones y Protestas")** y el **Tema 8 ("Negaciones e Interacciones")** serían muy activos en estas comunidades, reflejando descontento o movilización.
- **Comunidades "Neutras/Interesadas en Noticias":** Podrían agrupar a usuarios que discuten una gama más amplia de temas políticos como el **Tema 0 ("Guatemala")**, el **Tema 5 ("Delincuencia")** y el **Tema 9 ("Dinero")**, buscando información o debatiendo políticas públicas.

10. Interpretación y Contexto

0.1 Contextualización de los hallazgos

El análisis realizado permite comprender cómo las dinámicas en redes sociales influyen en la construcción de la opinión pública. En primer lugar, se observa que la red en torno a **@traficoGT** presenta una **alta centralización**: este usuario concentra la mayor parte de las interacciones (menciones y respuestas), lo que lo posiciona como el nodo más influyente y con mayor capacidad de difusión. Esto coincide con las métricas de centralidad, donde @traficoGT sobresale tanto en grado como en intermediación y cercanía, confirmando su rol como **puente principal de información**.

La red presenta una **densidad baja**, lo que indica que no todos los usuarios interactúan directamente entre sí, pero al mismo tiempo muestra propiedades de **mundo pequeño**: cualquier usuario puede conectarse con otro a través de pocos intermediarios. Asimismo, el coeficiente de agrupamiento revela la existencia de **clústeres moderadamente cohesionados**, lo que facilita la conformación de comunidades temáticas y la amplificación de mensajes en espacios reducidos.

La aplicación del algoritmo **Louvain** permitió identificar comunidades bien definidas, cada una con hashtags y temas predominantes. Estas comunidades reflejan intereses comunes, como el tráfico, la coyuntura política o las protestas ciudadanas. Dichos clústeres actúan como **cámaras de resonancia**, reforzando posturas y contribuyendo a la polarización o al consenso según la dinámica interna.

En cuanto al contenido, los **hashtags más usados** (#ahora, #guatemala, #urgente, entre otros) muestran la relevancia de la coyuntura nacional y la inmediatez de la información. Además, el análisis de sentimiento reveló una clara predominancia de mensajes **negativos y neutrales**, lo que refleja descontento ciudadano y una percepción crítica hacia las instituciones y actores políticos mencionados.

Finalmente, el estudio evidencia que los **influencers y comunidades digitales** no solo difunden información, sino que **estructuran la conversación pública**, legitiman narrativas y condicionan la percepción colectiva. La combinación de nodos influyentes, clústeres temáticos y emociones predominantes conforma un ecosistema donde la opinión pública se forma y transforma de manera constante, demostrando el poder estratégico de las redes sociales en la construcción del discurso social.