

# Social Network Analysis

## Authors:

- Adrian Flores
  - Andrea Ramirez
- 

## (1) Import Libraries

```
In [1]: #!pip install unidecode
```

```
Requirement already satisfied: unidecode in /usr/local/lib/python3.10/dist-packages  
(1.3.8)
```

```
In [2]: #!pip install fuzzywuzzy
```

```
Requirement already satisfied: fuzzywuzzy in /usr/local/lib/python3.10/dist-packages  
(0.18.0)
```

```
In [3]: # Data manipulation and visualization  
import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
import seaborn as sns  
from unidecode import unidecode  
import json  
import nltk  
from nltk.stem import SnowballStemmer  
from nltk.corpus import stopwords  
from fuzzywuzzy import process, fuzz  
from wordcloud import WordCloud  
import networkx as nx  
  
# Standard Libraries  
import warnings  
warnings.filterwarnings('ignore')
```

```
# ===== Reproducibility Seed =====  
# Set a fixed seed for the random number generator for reproducibility  
random_state = 42
```

```
# Set matplotlib inline  
%matplotlib inline
```

```
# Set default figure size  
plt.rcParams['figure.figsize'] = (6, 4)
```

```
# Define custom color palette  
palette = sns.color_palette("viridis", 12)
```

```
# Set the style of seaborn
sns.set(style="whitegrid")
```

```
/usr/local/lib/python3.10/dist-packages/fuzzywuzzy/fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
    warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
```

## (2) Data Upload

```
In [4]: # Read the .txt file
with open('tweets.txt', 'r', encoding='utf-16') as file:
    lines = file.readlines()
```

```
In [5]: # List to hold extracted tweet data
tweet_data = []

# Loop through each line, assuming each line is a separate JSON object
for line in lines:
    # Parse the line as JSON
    data = json.loads(line.strip())

    # Extract the relevant information
    tweet_entry = {
        "tweet_date": pd.to_datetime(data["date"]).date(),
        "user_username": data["user"]["username"],
        "mentioned_users": ", ".join([user["username"] for user in data["mentionedU
        "reply_count": data["replyCount"],
        "retweet_count": data["retweetCount"],
        "like_count": data["likeCount"],
        "quote_count": data["quoteCount"],
        'tweet': data['rawContent'],
        "hashtags": ", ".join(data["hashtags"])
    }

    # Append the tweet data to the list
    tweet_data.append(tweet_entry)
```

```
In [6]: # Create the DataFrame
df = pd.DataFrame(tweet_data)

# Display the DataFrame
df.head(3)
```

```
Out[6]: tweet_date user_username mentioned_users reply_count retweet_count like_count quote_count
```

	tweet_date	user_username	mentioned_users	reply_count	retweet_count	like_count	quote_count
0	2024-09-12	La_ReVoluzzion	usembassyguate, 48CantonesToto, USAIDGuate, UE...	0	0	0	0
1	2024-09-12	XelaNewsGt		12	80	142	
2	2024-09-12	M24095273	IvanDuque, BArevalodeLeon	0	0	0	

### Observaciones💡 -->

Notar que ya se han llevado a cabo algunas acciones importantes de pre-procesamiento con el objetivo de optimizar el conjunto de datos, en el contexto del análisis de redes sociales (esto puede variar según el propósito del dataset). A continuación, se detallan los pasos realizados:

- **Reducción de columnas:** Se han eliminado aquellas columnas que no aportan valor relevante para el análisis de redes sociales, dejando únicamente las siguientes variables: tweet\_date, user\_username, mentioned\_users, reply\_count, retweet\_count, like\_count, quote\_count, tweet y hashtags. La finalidad es reducir la dimensionalidad del conjunto de datos, eliminando características que solo añaden ruido innecesario.
- **Formateo de fecha:** La columna tweet\_date, que originalmente incluía tanto la fecha como la hora, ha sido simplificada para contener únicamente el día, mes y año. Este ajuste es especialmente útil dado que la precisión horaria no es relevante para el análisis, contribuyendo así a la homogeneización del formato de los datos.

## (3) Exploratory Analysis 🔎

### (1) Descripción General de los Datos

```
In [7]: # Print the number of records in the DataFrame  
print("The given dataset has", df.shape[0], "registers and", df.shape[1], "columns.")
```

The given dataset has 5019 registers and 9 columns.

### Observaciones💡 -->

- El conjunto de datos actual cuenta con 5019 registros y las 9 columnas previamente mencionadas, indicando una dimensión relativamente pequeña. Este conjunto de datos consta de tweets de la plataforma X, que de alguna u otra forma están ligados a el usuario @BArevalodeLeon . Cada uno de los 7613 registros representa un tweet único, mientras que las 9 columnas corresponden a diferentes características o variables medidas para cada observación, incluyendo el texto y fecha de un tweet, así como las estadísticas de este.

```
In [8]: # Basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5019 entries, 0 to 5018
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   tweet_date       5019 non-null    object  
 1   user_username     5019 non-null    object  
 2   mentioned_users  5019 non-null    object  
 3   reply_count      5019 non-null    int64  
 4   retweet_count    5019 non-null    int64  
 5   like_count       5019 non-null    int64  
 6   quote_count      5019 non-null    int64  
 7   tweet            5019 non-null    object  
 8   hashtags         5019 non-null    object  
dtypes: int64(4), object(5)
memory usage: 353.0+ KB
```

### Observaciones💡 -->

- Por el momento, no se ha detectado la presencia de valores faltantes en ninguna de las columnas. Sin embargo, es evidente que sí existe información ausente en algunos registros, lo que sugiere que será necesario realizar un preprocesamiento adicional para identificar adecuadamente estos datos faltantes. Una vez detectados, será importante aplicar técnicas de imputación que permitan completar o manejar estos valores de manera adecuada.

El conjunto de datos contiene 9 columnas o características, las cuales se describen a continuación:

- **tweet\_date**: La fecha en la que se publicó el tweet.
- **user\_username**: El nombre de usuario de la cuenta que publicó el tweet.
- **mentioned\_users**: Los usuarios mencionados en el tweet.
- **reply\_count**: El número de respuestas que recibió el tweet.
- **retweet\_count**: La cantidad de veces que el tweet fue retuiteado.

- **like\_count**: El número de "me gusta" que recibió el tweet.
- **quote\_count**: La cantidad de veces que el tweet fue citado.
- **tweet**: El contenido textual del tweet.
- **hashtags**: Los hashtags incluidos en el tweet.

Las características incluyen tanto datos de tipo numérico ( `int64` ) como de tipo texto ( `object` ).

## (2) Clasificación de las Variables

Nombre de la columna	Descripción	Clasificación
<code>tweet_date</code>	Fecha en la que se publicó el tweet	Cualitativa (descriptiva)
<code>user_username</code>	Nombre de usuario de la cuenta que publicó el tweet	Cualitativa (descriptiva)
<code>mentioned_users</code>	Usuarios mencionados en el tweet	Cualitativa (descriptiva)
<code>reply_count</code>	Número de respuestas que recibió el tweet	Cuantitativa (discreta)
<code>retweet_count</code>	Número de veces que fue retuiteado el tweet	Cuantitativa (discreta)
<code>like_count</code>	Número de "me gusta" que recibió el tweet	Cuantitativa (discreta)
<code>quote_count</code>	Número de veces que fue citado el tweet	Cuantitativa (discreta)
<code>tweet</code>	Contenido textual del tweet	Cualitativa (descriptiva)
<code>hashtags</code>	Hashtags incluidos en el tweet	Cualitativa (descriptiva)

### Observaciones -->

- El conjunto de datos posee 5 variables cualitativas descriptivas.
- Las últimas 4 variables del conjunto de datos son de tipo cuantitativo.

## (3) Exploración y Limpieza Inicial de los Datos

### (1) Preprocesamiento de los Datos

```
In [9]: # Download the NLTK stopwords if not already available
nltk.download('stopwords')
# Initialize the PorterStemmer
stemmer = SnowballStemmer('spanish')
```

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

```
In [10]: # Get the list of stopwords from NLTK
stop_words = set(stopwords.words('spanish'))
```

```
In [11]: # Function to remove stopwords and apply stemming
def preprocess_text(text):
    # Tokenize the text
    words = text.split()
    # Remove stopwords and apply stemming
    processed_words = [word for word in words if word not in stop_words.union({'si'})]
    # Reassemble the text
    return ' '.join(processed_words)
```

**Columna tweet -->**

```
In [12]: # Convert all entries to strings
df['tweet'] = df['tweet'].astype(str)
# Remove Leading/trailing whitespaces
df['tweet'] = df['tweet'].str.strip()
# Remove accents and/or special characters
df['tweet'] = df['tweet'].apply(unidecode)
# Convert to Lowercase
df['tweet'] = df['tweet'].str.lower()
# Remove usernames from the tweet text
df['tweet'] = df['tweet'].str.replace(r'@\w+', '', regex=True).str.strip()
# Remove URLs
df['tweet'] = df['tweet'].str.replace(r'http\S+|www\S+|https\S+', '', case=False, regex=True)
# Keeping letters, numbers, and spaces
df['tweet'] = df['tweet'].str.replace(r'[^w\s]', '', regex=True)
# Remove extra spaces
df['tweet'] = df['tweet'].str.replace(r'\s+', ' ', regex=True)
# Apply preprocessing (stopwords removal and stemming)
df['tweet'] = df['tweet'].apply(preprocess_text)
```

**Columna hashtags -->**

```
In [13]: # Convert all entries to strings
df['hashtags'] = df['hashtags'].astype(str)
# Remove Leading/trailing whitespaces
df['hashtags'] = df['hashtags'].str.strip()
# Remove accents and/or special characters
df['hashtags'] = df['hashtags'].apply(unidecode)
# Convert to Lowercase
df['hashtags'] = df['hashtags'].str.lower()
# Remove extra spaces
df['hashtags'] = df['hashtags'].str.replace(r'\s+', ' ', regex=True)
```

**Observaciones💡 -->**

A continuación, se describen los pasos realizados y sus justificaciones:

1. **Conversión a Cadena de Texto:** Asegura que todas las entradas sean tratadas de manera uniforme, evitando errores en operaciones posteriores.
2. **Eliminación de Espacios en Blanco:** Previene inconsistencias que podrían afectar el análisis, garantizando que cada entrada esté limpia.
3. **Eliminación de Acentos y Caracteres Especiales:** Normaliza los caracteres, facilitando la comparación entre palabras que pueden tener variantes acentuadas.
4. **Conversión a Minúsculas:** Reduce la variabilidad en los datos, asegurando que palabras como "Hola" y "hola" sean tratadas de manera consistente.
5. **Eliminación de Nombres de Usuario:** Permite centrarse en el contenido del mensaje en lugar de las interacciones entre usuarios, ya que tenemos estas interacciones almacenadas ya en otra columna.
6. **Eliminación de URLs:** Evita incluir enlaces que no aportan valor al análisis de sentimiento o contenido, enfocándose en el texto relevante.
7. **Eliminación de Caracteres Especiales:** Mantiene solo letras, números y espacios, eliminando el ruido que podría afectar la calidad del análisis.
8. **Eliminación de Espacios Adicionales:** Asegura que el texto final esté bien formado, facilitando la tokenización y otras operaciones de procesamiento.
9. **Aplicación de Preprocesamiento Adicional:** Incluye la eliminación de palabras vacías y la reducción a la raíz de las palabras, concentrándose en términos significativos y reduciendo la dimensionalidad del texto.

**Nota:**

1. La eliminación de **stopwords** filtra palabras comunes que, aunque frecuentes, aportan poco valor semántico al análisis, como "y", "el", "en", entre otras. Al excluir estas palabras, se enfoca el modelo en términos más significativos, lo que puede resultar en una mejora notable en la precisión y eficiencia del análisis textual. Sin embargo, el beneficio más grande es la reducción de dimensionalidad, permitiendo que el entrenamiento sea más rápido. [\[Referencia\]](#)
2. Para algunas de las tareas de procesamiento de lenguaje natural descritas con anterioridad, se optó por implementar nltk, para más información por favor ingresar a la documentación oficial en el siguiente [enlace](#).

**Valores Faltantes -->**

In [14]: `# Replace entries that are empty or contain only whitespace with np.nan`

```
df = df.replace(r'^\s*$', np.nan, regex=True)
```

## Observaciones💡 -->

- Esto nos permite identificar con precisión los valores faltantes y facilita la realización de imputaciones adecuadas en etapas posteriores, en caso de que sea necesario.

```
In [15]: df.isnull().sum()
```

```
Out[15]:
```

	0
tweet_date	0
user_username	0
mentioned_users	183
reply_count	0
retweet_count	0
like_count	0
quote_count	0
tweet	365
hashtags	4771

**dtype:** int64

```
In [16]: # Calculate the percentage of null values in each column
null_percentage = (df.isnull().sum() / len(df)) * 100

# Sort the null_percentage Series from highest to lowest
null_percentage_sorted = null_percentage.sort_values(ascending=False)

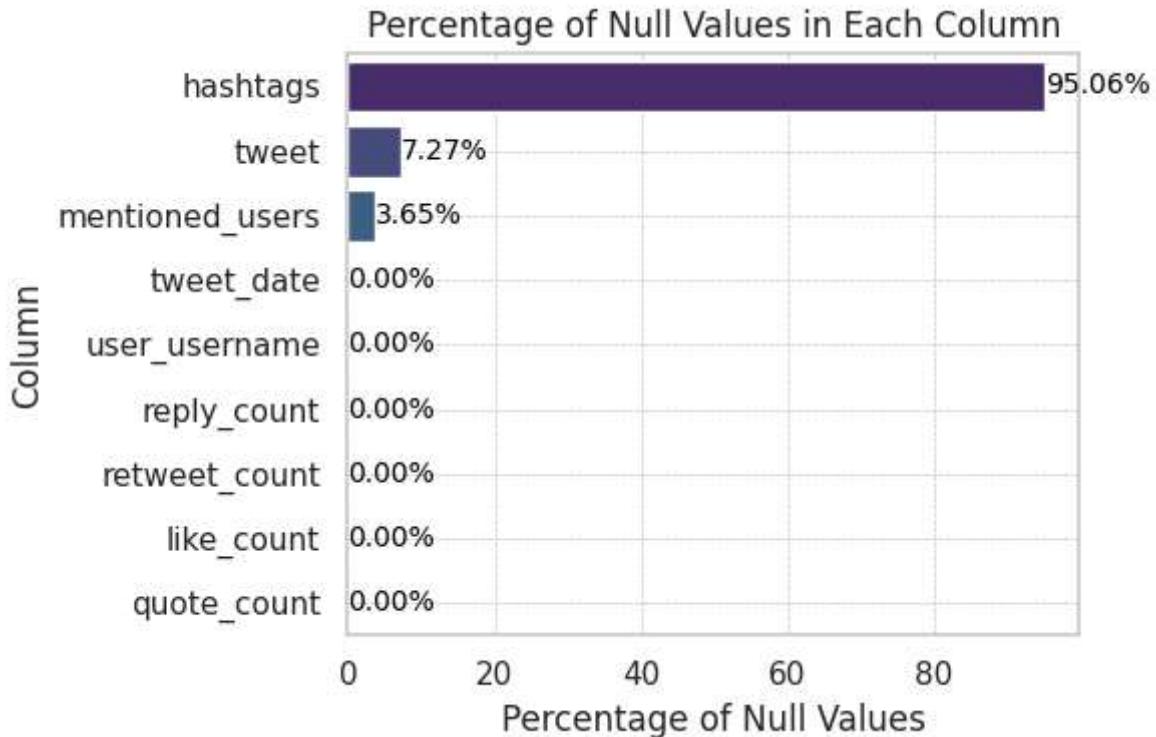
# Create a horizontal bar chart
ax = sns.barplot(x=null_percentage_sorted.values, y=null_percentage_sorted.index, p

# Add percentage in each bar
for bar in ax.patches:
    width = bar.get_width()
    ax.text(width, bar.get_y() + bar.get_height() / 2,
            '{:.2f}%'.format(width),
            ha='left', va='center', color='black', fontsize=10)

plt.xlabel("Percentage of Null Values") # Updated Label
plt.ylabel("Column") # Updated Label
plt.title("Percentage of Null Values in Each Column") # Fixed closing quote

# Adding grid with custom style
```

```
plt.grid(True, linestyle='--', linewidth=0.5) # Adding grid with dashed Lines and  
  
# Show the plot  
plt.tight_layout() # Adjust Layout  
plt.show()
```



### Observaciones💡 -->

- Observamos que la columna hashtags presenta más del 95% de datos faltantes. Aunque esto sugiere que podría no ser útil, en lugar de eliminarla, considero que podríamos imputar los valores o analizar la posibilidad de utilizarla para extraer información adicional. Esto podría enriquecer nuestro análisis sin perder la columna por completo.

```
In [17]: # Replace entries that are empty or contain only whitespace with np.nan  
df = df.replace(np.nan, 'None')
```

## (2) Exploración de los Datos

### (1) ¿Cómo se distribuye la longitud de los tweets y qué tendencias podemos identificar?

```
In [18]: # Calculate the length of text entries in the 'text' column.  
length = df["tweet"].apply(len)  
# Display descriptive statistics of text lengths.  
print("President Tweet Set: Tweet Length Statistics")  
print(length.describe())
```

```
President Tweet Set: Tweet Length Statistics
count      5019.000000
mean       72.588563
std        68.695351
min        1.000000
25%       25.000000
50%       55.000000
75%      107.000000
max      1542.000000
Name: tweet, dtype: float64
```

### Observaciones💡 -->

- La longitud promedio de los tweets es de aproximadamente 73 caracteres, lo que sugiere que la mayoría de los tweets son relativamente cortos.
- La desviación estándar es de aproximadamente 68.88, lo que indica una variabilidad considerable en la longitud de los tweets. Esto sugiere que algunos tweets son mucho más largos o más cortos que la media, lo que puede influir en la forma en que se interpretan.
- Al analizar las longitudes extremas, se observa que el tweet más corto tiene solo 1 carácter, mientras que el más largo alcanza los 1542 caracteres. Esta variabilidad extrema resalta que, aunque la mayoría de los tweets son breves, existen excepciones significativas que pueden contener información valiosa o contexto adicional.

### (2) ¿Cómo se distribuyen las estadísticas de los tweets y qué tendencias podemos identificar?

```
In [19]: numeric = df.select_dtypes(include='number')
numeric.describe()
```

```
Out[19]:   reply_count  retweet_count  like_count  quote_count
count      5019.000000      5019.000000      5019.000000      5019.000000
mean       7.408249       21.775254     109.174935      1.780634
std        110.363737      279.448721    1869.974212      25.948332
min        0.000000       0.000000       0.000000       0.000000
25%       0.000000       0.000000       0.000000       0.000000
50%       0.000000       0.000000       0.000000       0.000000
75%       0.000000       0.000000      2.000000       0.000000
max      4783.000000      8307.000000    67416.000000     1336.000000
```

## Observaciones -->

- El promedio de respuestas es de 7.41, mientras que el de retweets es de 21.78 y el de likes asciende a 109.17. El número promedio de citas es de 1.78. Sin embargo, estos promedios son engañosos debido a la alta desviación estándar y la distribución sesgada, lo que sugiere que solo una pequeña parte de los tweets genera una cantidad significativa de interacciones. La mayoría de los tweets probablemente están por debajo de estos valores.
- La desviación estándar es considerablemente alta en todas las categorías, especialmente en likes (1869.97) y retweets (279.45). Esto indica una gran dispersión de los datos, con algunos tweets obteniendo miles de interacciones, mientras que muchos otros obtienen muy pocas o ninguna.
- La mediana en todas las categorías es 0, lo que significa que más del 50% de los tweets no reciben ninguna respuesta, retweet, like o cita. Esto confirma que, aunque algunos tweets obtienen una cantidad significativa de interacciones, la mayoría no genera ningún tipo de respuesta por parte de los usuarios.
- Las interacciones máximas muestran algunos valores extremadamente altos, como 4783 respuestas, 8307 retweets, 67416 likes y 1336 citas en un solo tweet. Estos valores sugieren que algunos tweets excepcionales lograron una viralidad considerable, lo que elevó los promedios y amplió la desviación estándar.

### (3) ¿Cuáles son los hashtags más utilizados y qué patrones se pueden observar en su uso?

```
In [20]: from collections import Counter

# Split the hashtags by comma and flatten the list
all_hashtags = df['hashtags'].str.cat(sep=',').split(',')
all_hashtags = [hashtag.strip() for hashtag in all_hashtags if hashtag.strip().lower()

# Step 2: Count the frequency of each hashtag
hashtag_counts = Counter(all_hashtags)

# Step 3: Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(hashtag_counts)

# Plot the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off') # Turn off the axis
plt.title("Hashtag Word Cloud", fontsize=16)
plt.show()
```

## Hashtag Word Cloud



### Observaciones -->

- Los hashtags más frecuentes en el conjunto de datos incluyen #GuatemalaSaleAdelante, #Urgente, #MinFinSaleAdelante, y #Presupuesto2025, entre otros. Estos hashtags no solo reflejan temas de interés nacional y política fiscal, sino también campañas o movimientos impulsados por el gobierno o medios de comunicación. Su popularidad sugiere que el contenido está vinculado a eventos actuales y temas de relevancia social y económica en Guatemala.

### (4) ¿Cuáles son las palabras más utilizadas en los tweets y qué patrones se pueden observar en su uso?

In [21]:

```
# Split the tweets by comma and flatten the list
all_words = df['tweet'].str.cat(sep=' ').split(' ')

# Step 2: Count the frequency of each hashtag
word_counts = Counter(all_words)

# Step 3: Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_fro

# Plot the word cloud
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off') # Turn off the axis
plt.title("Hashtag Word Cloud", fontsize=16)
plt.show()
```



## (4) Topology Analysis & Metrics

```
In [22]: # Convert 'mentioned_users' into a list of users (assuming users are separated by comma)
df['mentioned_users'] = df['mentioned_users'].apply(lambda x: x.split(',') if pd.notnull(x) else [])

# Step 1: Create a directed graph
G = nx.DiGraph()

# Add edges to the graph from user_username to mentioned_users
for index, row in df.iterrows():
    source_user = row['user_username']
    mentioned_users = row['mentioned_users']

    for target_user in mentioned_users:
        G.add_edge(source_user, target_user)

# Step 2: Identify the most connected nodes
degree_dict = dict(G.degree(G.nodes())) # Degree of each node (sum of in-degree and out-degree)
sorted_degree = sorted(degree_dict.items(), key=lambda x: x[1], reverse=True)

# Display top 5 most connected nodes
print("Top 5 most connected nodes:")
for node, degree in sorted_degree[:5]:
    print(f"{node}: {degree} connections")

# Step 3: Visualize the Graph
pos = nx.spring_layout(G) # Layout for a visually appealing graph

# Draw the nodes, edges, and labels
nx.draw(G, pos, with_labels=True, node_color='lightblue', edge_color='gray', node_size=1000)

# Highlight the top connected nodes
top_nodes = [node for node, degree in sorted_degree[:5]]
nx.draw_networkx_nodes(G, pos, nodelist=top_nodes, node_color='red', node_size=700)

plt.title("User Interaction Network")
plt.show()
```

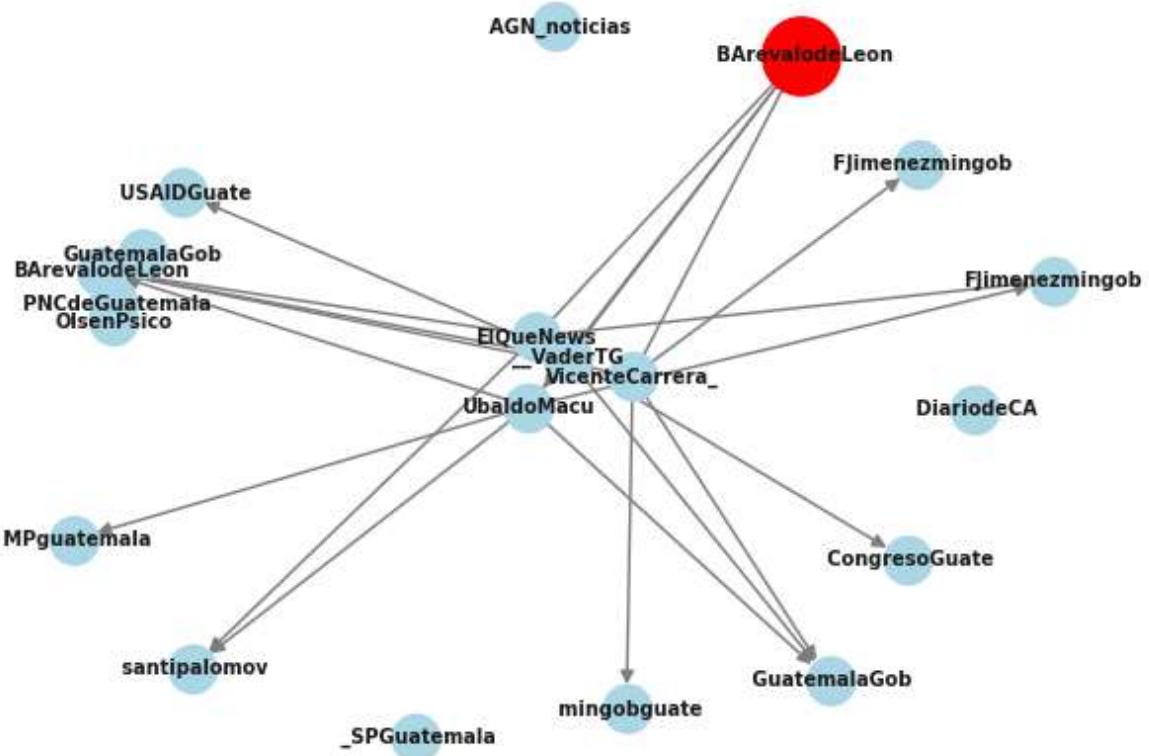
```
Top 5 most connected nodes:  
BArevalodeLeon: 2072 connections  
BArevalodeLeon: 754 connections  
GuatemalaGob: 480 connections  
UbaldoMacu: 382 connections  
santipalomov: 359 connections
```

User Interaction Network



```
In [23]: # Step 1: Filter the top most connected users (e.g., top 10 by degree)  
top_n = 20 # You can adjust this to any number of top users you'd like to display  
important_users = [node for node, degree in sorted_degree[:top_n]]  
  
# Create a subgraph with just the most connected users and their interactions  
important_subgraph = G.subgraph(important_users)  
  
# Step 2: Visualize the subgraph  
pos = nx.spring_layout(important_subgraph) # Layout for positioning the nodes  
  
# Draw nodes and edges for the most important users  
nx.draw(important_subgraph, pos, with_labels=True, node_color='lightblue', edge_col  
  
# Highlight the top user with a distinct color (optional)  
nx.draw_networkx_nodes(important_subgraph, pos, nodelist=[important_users[0]], node  
  
plt.title(f"Top {top_n} Most Connected Users")  
plt.show()
```

## Top 20 Most Connected Users



```
In [24]: # 1. Network Density
network_density = nx.density(G)
print(f"Network Density: {network_density:.4f}")

# 2. Network Diameter for Directed Graphs
if nx.is_strongly_connected(G):
    # For strongly connected graphs
    network_diameter = nx.diameter(G.to_undirected()) # Convert to undirected for
    print(f"Network Diameter (Strongly Connected): {network_diameter}")
else:
    # For weakly connected graphs
    weakly_connected_components = list(nx.weakly_connected_components(G))
    diameters = [nx.diameter(G.subgraph(component).to_undirected()) for component in weakly_connected_components]
    if diameters:
        print(f"Network Diameter (Weakly Connected): {max(diameters)}")
    else:
        print("No valid diameter found in weakly connected components.")

# 3. Clustering Coefficient
average_clustering = nx.average_clustering(G.to_undirected()) # Use undirected version
print(f"Average Clustering Coefficient: {average_clustering:.4f}")
```

Network Density: 0.0011  
Network Diameter (Weakly Connected): 6  
Average Clustering Coefficient: 0.1837

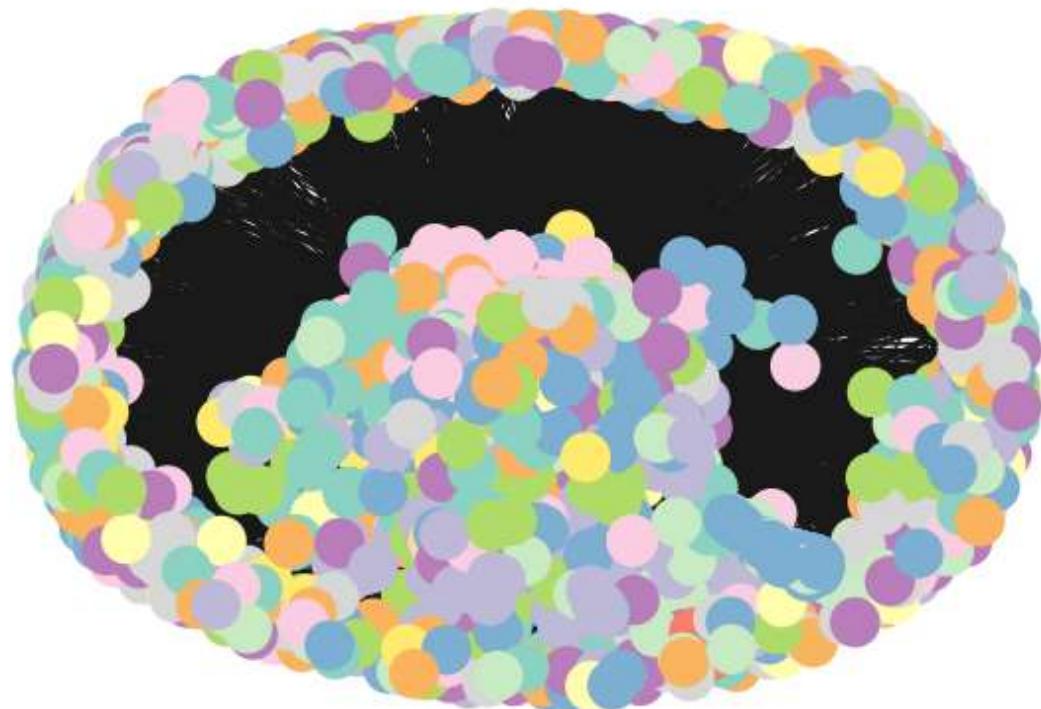
## (5) Community Identification & Analysis

## (1) Algoritmo a utilizar

Los dos algoritmos más utilizados para la detección de comunidades en grafos **dirigidos** son Louvain y Girvan-Newman. Aunque ambos son populares, la principal diferencia entre ellos radica en que el algoritmo de Louvain es significativamente más rápido, lo que lo hace ideal para grafos grandes. Por otro lado, el algoritmo de Girvan-Newman es más robusto, pero su tiempo de ejecución aumenta considerablemente en grafos de gran tamaño. Por ello, Louvain suele ser la mejor opción para esta aplicación debido a su rapidez y versatilidad.

## (2) Detección de comunidades

```
In [25]: import community.community_louvain as community_louvain  
  
In [26]: partition = community_louvain.best_partition(G.to_undirected())  
  
In [27]: colors = [partition[node] for node in G.nodes()]  
  
# Draw the graph  
pos = nx.spring_layout(G)  
nx.draw(G, pos, node_color=colors, with_labels=False, cmap=plt.cm.Set3)  
plt.show()
```



Las 3 comunidades más grandes -->

```
In [28]: # Count the size of each community
community_sizes = Counter(partition.values())

# Get the top 3 Largest communities
top_communities = [community for community, size in community_sizes.most_common(3)]

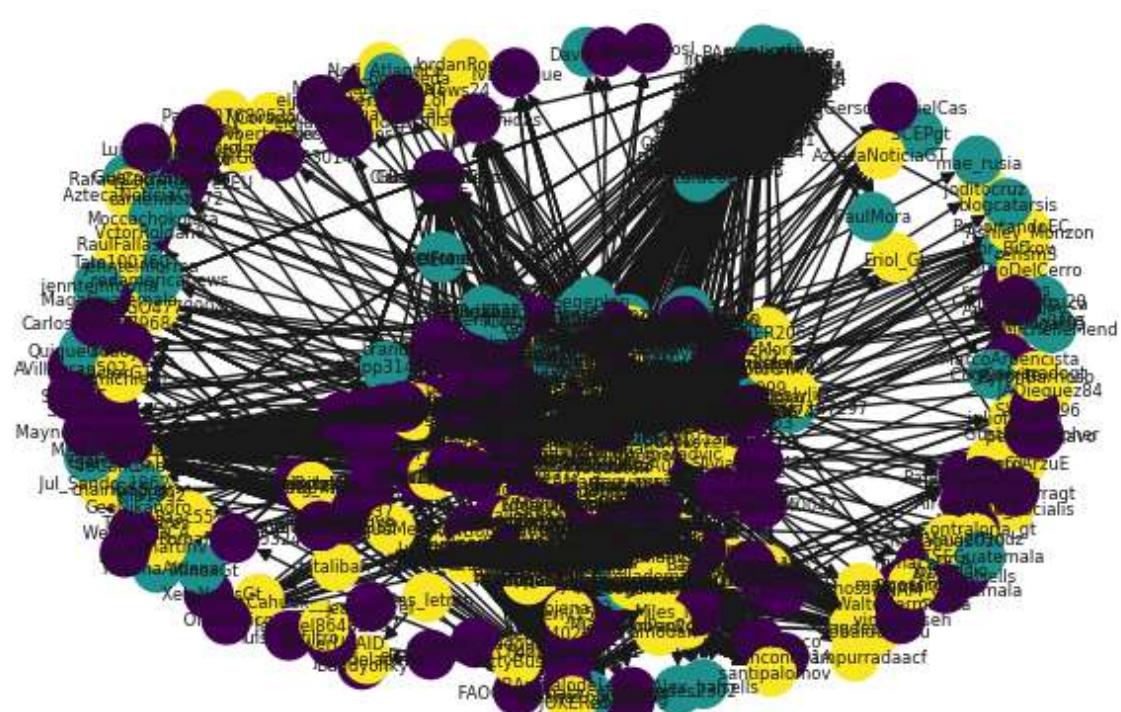
# Filter nodes that are in the top 3 communities
top_nodes = [node for node in G.nodes() if partition[node] in top_communities]

# Create a subgraph containing only the top 3 communities
subgraph = G.subgraph(top_nodes)

# Map communities to colors
from matplotlib import cm
colormap = cm.get_cmap('viridis', 3)
community_to_color = {top_communities[i]: colormap(i) for i in range(3)}

# Assign colors to nodes based on their community
colors = [community_to_color[partition[node]] for node in subgraph.nodes()]

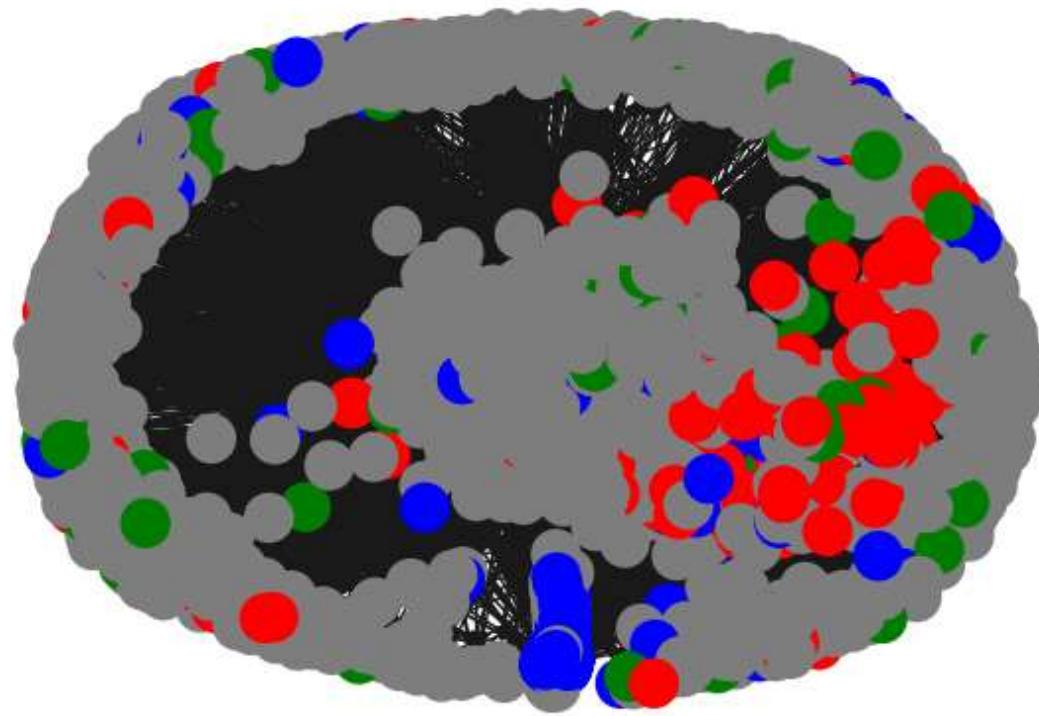
# Draw the graph
pos = nx.spring_layout(subgraph)
nx.draw(subgraph, pos, node_color=colors, with_labels=True, font_size=6, cmap=plt.cm.viridis)
plt.show()
```



**Todos los nodos con las comunidades más grandes resaltadas -->**

```
In [31]: highlight_colors = {top_communities[0]: 'red', top_communities[1]: 'blue', top_communities[2]: 'green'}
colors = [highlight_colors.get(partition[node], 'gray') for node in G.nodes()]
```

```
# Draw the graph with the custom colors
pos = nx.spring_layout(G)
nx.draw(G, pos, node_color=colors, with_labels=False, cmap=plt.cm.Set3)
plt.show()
```



### Nodos más influyentes en cada comunidad -->

```
In [29]: # Filter nodes that are in the top 3 communities
top_nodes = [node for node in G.nodes() if partition[node] in top_communities]

# Create a subgraph containing only the top 3 communities
subgraph = G.subgraph(top_nodes)

# Calculate degree centrality for each node (can use in_degree and out_degree for d
in_degrees = dict(subgraph.in_degree()) # In-degree for incoming edges
total_degrees = {node: in_degrees[node] for node in subgraph.nodes()} # Total degr

# Identify the most prominent nodes in each community
prominent_nodes_by_community = {}

for community in top_communities:
    # Get nodes belonging to the current community
    community_nodes = [node for node in subgraph.nodes() if partition[node] == commu

    # Sort nodes by their total degree (connections) and get the top node(s)
    sorted_nodes = sorted(community_nodes, key=lambda node: total_degrees[node], re

    # Store the top node(s) for this community
    prominent_nodes_by_community[community] = sorted_nodes[:5] # Get top 3 most co

# Print out the most prominent nodes in each community
```

```

for community, nodes in prominent_nodes_by_community.items():
    print(f"Community {community} most prominent nodes:")
    for node in nodes:
        print(f"Node: {node}, Total Connections: {total_degrees[node]}")
    print()

```

Community 4 most prominent nodes:

Node: BARevalodeLeon, Total Connections: 902  
 Node: IvanDuque, Total Connections: 110  
 Node: OlsenPsico, Total Connections: 109  
 Node: GuatemalaGob, Total Connections: 105  
 Node: GersonGudielCas, Total Connections: 50

Community 7 most prominent nodes:

Node: BARevalodeLeon, Total Connections: 551  
 Node: PEdgar05, Total Connections: 10  
 Node: Alex\_balsells, Total Connections: 7  
 Node: arbencista1944, Total Connections: 5  
 Node: Ejercito\_GT, Total Connections: 4

Community 1 most prominent nodes:

Node: UbaldoMacu, Total Connections: 257  
 Node: santipalomov, Total Connections: 239  
 Node: MPguatemala, Total Connections: 214  
 Node: UbaldoMacu, Total Connections: 54  
 Node: Ashley\_Monzon, Total Connections: 18

```

In [30]: # For each community, create a word cloud
for community in top_communities:
    # Get the users in this community
    community_users = [node for node in subgraph.nodes() if partition[node] == community]

    # Filter the dataframe for these users
    community_tweets = df[df['user_username'].isin(community_users)]['tweet']

    # Concatenate all tweets into one large string
    all_tweets = " ".join(tweet for tweet in community_tweets)

    # Generate word cloud
    wordcloud = WordCloud(width=800, height=400, background_color="white", stopwords=stopwords)

    # Plot the word cloud
    plt.figure(figsize=(6, 4))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.title(f"Word Cloud for Community {community}")
    plt.show()

```

## Word Cloud for Community 4



## Word Cloud for Community 7



## Word Cloud for Community 1



## **Observaciones -->**

- Las tres comunidades más grandes, según la numeración asignada automáticamente por el paquete Networkx, son las comunidades 4, 1 y 7.

En las comunidades 4 y 7, la cuenta del presidente es el nodo más conectado, mientras que la comunidad 1 está formada por la cuenta del Ministerio Público, algunos funcionarios de la Secretaría de Comunicación Social de la Presidencia de Guatemala, y cuentas de usuarios regulares.

- El nodo más conectado en la comunidad 1 es 'ubaldomacu', quien no es una figura pública ni un funcionario de gobierno, sino un usuario regular de la plataforma que participa activamente en las discusiones.
- Las nubes de palabras de cada comunidad muestran varias palabras en común, siendo las más repetidas 'presidente' y 'corrupto'. Sin embargo, la comunidad 1 se destaca por el uso frecuente de términos como 'periodista' y 'hetcenter', lo que sugiere que las discusiones entre los nodos de esta comunidad están relacionadas con reportajes periodísticos y el debate sobre el supuesto uso de netcenters para influir en conversaciones durante el período electoral.
- Las comunidades 4 y 7 se caracterizan por discusiones centradas en el gobierno y el presidente. La comunidad 4 adopta un tono más crítico y agresivo hacia el gobierno, mientras que la comunidad 7 muestra un enfoque más neutro, con opiniones mixtas entre positivas y negativas.

## (6) Influencer Analysis and Key Nodes

```
In [33]: # Node metrics
degree_centrality = nx.degree_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)
closeness_centrality = nx.closeness_centrality(G)

# Combine the results into a DataFrame
centrality_df = pd.DataFrame({
    'username': [node for node in G.nodes()],
    'degree_centrality': [degree_centrality[node] for node in G.nodes()],
    'betweenness_centrality': [betweenness_centrality[node] for node in G.nodes()],
    'closeness_centrality': [closeness_centrality[node] for node in G.nodes()]
})

# Sort by each metric
top_by_degree = centrality_df.sort_values(by='degree_centrality', ascending=False)
top_by_betweenness = centrality_df.sort_values(by='betweenness_centrality', ascending=False)
top_by_closeness = centrality_df.sort_values(by='closeness_centrality', ascending=False)

# Print the results
print("Top 10 Users by Degree Centrality:")
print(top_by_degree[['username', 'degree_centrality']])

print("\nTop 10 Users by Betweenness Centrality:")
print(top_by_betweenness[['username', 'betweenness_centrality']])

print("\nTop 10 Users by Closeness Centrality:")
print(top_by_closeness[['username', 'closeness_centrality']])
```

Top 10 Users by Degree Centrality:

	username	degree_centrality
5	BArevalodeLeon	0.615385
12	BArevalodeLeon	0.223938
49	GuatemalaGob	0.142560
70	UbaldoMacu	0.113454
238	santipalomov	0.106623
65	MPguatemala	0.098307
15	CongresoGuate	0.064152
99	mingobguate	0.061182
3	USAIDGuate	0.055242
25	GuatemalaGob	0.055242

Top 10 Users by Betweenness Centrality:

	username	betweenness_centrality
70	UbaldoMacu	0.011843
1096	marcoanmo72	0.009244
255	wichomarg	0.009239
1099	Anarchiaeth	0.008940
1412	Santillana87	0.008673
1349	KaddV	0.008554
115	Patrici07680625	0.005052
33	BenitoC67601310	0.004519
1247	RafaelAE1967	0.003964
164	__VaderTG	0.003956

Top 10 Users by Closeness Centrality:

	username	closeness_centrality
5	BArevalodeLeon	0.615385
12	BArevalodeLeon	0.379872
7	None	0.306409
49	GuatemalaGob	0.200021
97	FJimenezmingob	0.147744
238	santipalomov	0.146731
70	UbaldoMacu	0.130547
65	MPguatemala	0.123838
99	mingobguate	0.120240
25	GuatemalaGob	0.107920

### Observaciones -->

- **Degree Centrality** mide cuántas conexiones directas tiene un nodo, lo que refleja su nivel de actividad o influencia directa. En este análisis, los nodos más influyentes según esta métrica corresponden al presidente y a cuentas gubernamentales como USAID, el Ministerio Público, el Congreso y el Ministerio de Gobernación. Entre estas cuentas oficiales, destaca la cuenta de 'UbaldoMacu', que pertenece a un usuario regular de la plataforma.
- **Betweenness Centrality** indica cuántas veces un nodo actúa como puente o intermediario en los caminos más cortos entre otros nodos. Los nodos con los valores más altos en esta métrica corresponden a varios

usuarios comunes de la plataforma, quienes participan activamente en diversas discusiones relacionadas con el tema principal de esta red: el presidente Bernardo Arévalo. La aparición de estos usuarios como los más influyentes en esta métrica es de esperarse, ya que participan en una gran variedad de discusiones entre varios usuarios, por lo que, naturalmente, actuarán como un 'puente' entre varios nodos.

- **Closeness Centrality** mide qué tan cerca está un nodo de todos los demás en el grafo, lo que refleja la facilidad con la que puede acceder o difundir información en la red. Según esta métrica, los nodos más influyentes nuevamente incluyen la cuenta del presidente y otras cuentas gubernamentales. Además, destaca nuevamente 'UbaldoMacu', junto a 'santipalomov', secretario de la Secretaría de Comunicación Social de la Presidencia de Guatemala, quien también apareció en puntos anteriores, como en la detección de comunidades.

## (7) Isolated Groups

Para la identificación de grupos aislados, se utilizará el método

`weakly_connected_components`, el cual encuentra grupos de nodos conectados entre sí, ignorando la dirección de la conexión. Este método es preferible sobre otros, ya que `strongly_connected_components` sí toma en cuenta la dirección, y esto genera una cantidad inmensa de grupos de un solo nodo. Por otro lado, `connected_components` está diseñado para grafos no dirigidos, y cumple una función similar a `weakly_connected_components`.

```
In [62]: in_degree_centrality = nx.in_degree_centrality(G)
          out_degree_centrality = nx.out_degree_centrality(G)
```

```
In [63]: def print_top_centralities(centrality_dict, centrality_name, nodes, top_n=None):
    # Filter the centralities to only those present in the provided nodes
    filtered_centrality = {node: centrality_dict[node] for node in nodes if node in
    # Sort and limit the output based on top_n
    sorted_centrality = sorted(filtered_centrality.items(), key=lambda x: x[1], reverse=True)
    if top_n:
        sorted_centrality = sorted_centrality[:top_n]

    print(f"\n{centrality_name} for the users in the group:")
    for user, value in sorted_centrality:
        print(f"User: {user}, {centrality_name}: {value:.4f}")
```

```
In [67]: # Weakly connected components
          weakly_connected_components = list(nx.weakly_connected_components(G))

          # Get the count of nodes and print the users in groups with fewer than 5 members
          for idx, component in enumerate(weakly_connected_components):
              component_size = len(component)
```

```
print(f"\nGroup {idx}: Size = {component_size}")

# If the component is small, list the users
if component_size < 5:
    print(f"Users in small group {idx}: {list(component)}")

# Print centralities for the users in this group
print_top_centralities(in_degree_centrality, "In-Degree Centrality", component)
print_top_centralities(out_degree_centrality, "Out-Degree Centrality", component)
print_top_centralities(betweenness_centrality, "Betweenness Centrality", component)
print_top_centralities(closeness_centrality, "Closeness Centrality", component)

else:
    print("^\nMain group")
```

Group 0: Size = 3359  
^ Main group

Group 1: Size = 2  
Users in small group 1: ['inafinogenova', 'LaBase\_TV']

In-Degree Centrality for the users in the group:  
User: inafinogenova, In-Degree Centrality: 0.0003  
User: LaBase\_TV, In-Degree Centrality: 0.0000

Out-Degree Centrality for the users in the group:  
User: LaBase\_TV, Out-Degree Centrality: 0.0003  
User: inafinogenova, Out-Degree Centrality: 0.0000

Betweenness Centrality for the users in the group:  
User: inafinogenova, Betweenness Centrality: 0.0000  
User: LaBase\_TV, Betweenness Centrality: 0.0000

Closeness Centrality for the users in the group:  
User: inafinogenova, Closeness Centrality: 0.0003  
User: LaBase\_TV, Closeness Centrality: 0.0000

Group 2: Size = 2  
Users in small group 2: ['OurWorldInData', '\_HannahRitchie']

In-Degree Centrality for the users in the group:  
User: \_HannahRitchie, In-Degree Centrality: 0.0003  
User: OurWorldInData, In-Degree Centrality: 0.0000

Out-Degree Centrality for the users in the group:  
User: OurWorldInData, Out-Degree Centrality: 0.0003  
User: \_HannahRitchie, Out-Degree Centrality: 0.0000

Betweenness Centrality for the users in the group:  
User: OurWorldInData, Betweenness Centrality: 0.0000  
User: \_HannahRitchie, Betweenness Centrality: 0.0000

Closeness Centrality for the users in the group:  
User: \_HannahRitchie, Closeness Centrality: 0.0003  
User: OurWorldInData, Closeness Centrality: 0.0000

Group 3: Size = 2  
Users in small group 3: ['fedefut\_oficial', 'CuxMigue']

In-Degree Centrality for the users in the group:  
User: fedefut\_oficial, In-Degree Centrality: 0.0003  
User: CuxMigue, In-Degree Centrality: 0.0000

Out-Degree Centrality for the users in the group:  
User: CuxMigue, Out-Degree Centrality: 0.0003  
User: fedefut\_oficial, Out-Degree Centrality: 0.0000

Betweenness Centrality for the users in the group:  
User: fedefut\_oficial, Betweenness Centrality: 0.0000  
User: CuxMigue, Betweenness Centrality: 0.0000

```
Closeness Centrality for the users in the group:  
User: fedefut_oficial, Closeness Centrality: 0.0003  
User: CuxMigue, Closeness Centrality: 0.0000  
  
Group 4: Size = 3  
Users in small group 4: ['SenadoGovCo', 'ISAZULETA', 'EfrainCepeda']
```

```
In-Degree Centrality for the users in the group:  
User: SenadoGovCo, In-Degree Centrality: 0.0003  
User: EfrainCepeda, In-Degree Centrality: 0.0003  
User: ISAZULETA, In-Degree Centrality: 0.0000
```

```
Out-Degree Centrality for the users in the group:  
User: ISAZULETA, Out-Degree Centrality: 0.0006  
User: SenadoGovCo, Out-Degree Centrality: 0.0000  
User: EfrainCepeda, Out-Degree Centrality: 0.0000
```

```
Betweenness Centrality for the users in the group:  
User: SenadoGovCo, Betweenness Centrality: 0.0000  
User: ISAZULETA, Betweenness Centrality: 0.0000  
User: EfrainCepeda, Betweenness Centrality: 0.0000
```

```
Closeness Centrality for the users in the group:  
User: SenadoGovCo, Closeness Centrality: 0.0003  
User: EfrainCepeda, Closeness Centrality: 0.0003  
User: ISAZULETA, Closeness Centrality: 0.0000
```

```
In [52]: # Generate Word Cloud for each group  
for i, component in enumerate(weakly_connected_components):  
    # Get usernames in the current component  
    usernames_in_group = list(component)  
  
    # Filter the dataframe for tweets of users in this group  
    group_df = df[df['user_username'].isin(usernames_in_group)]  
  
    # Concatenate all tweets into a single text string  
    all_tweets = " ".join(group_df['tweet'].tolist())  
  
    # Generate word cloud  
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_tweets)  
  
    # Display the word cloud  
    plt.figure(figsize=(5, 3))  
    plt.imshow(wordcloud, interpolation='bilinear')  
    plt.axis("off")  
    plt.title(f"Word Cloud for Group {i + 1} (Node Count: {len(component)})")  
    plt.show()
```

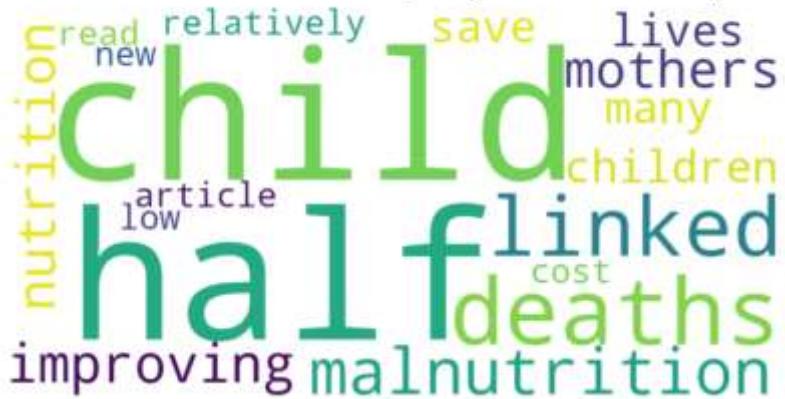
Word Cloud for Group 1 (Node Count: 3359)



Word Cloud for Group 2 (Node Count: 2)

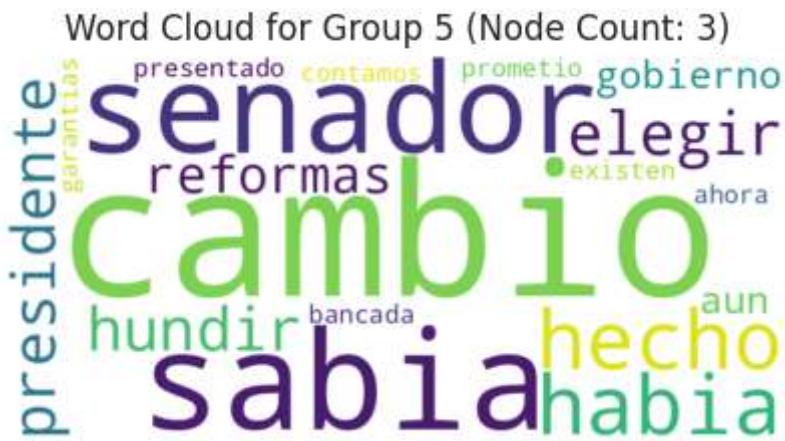


Word Cloud for Group 3 (Node Count: 2)



Word Cloud for Group 4 (Node Count: 2)





#### **Observaciones -->**

- Se identificaron cinco grupos débilmente conectados. Uno de ellos corresponde al cuerpo principal de la red, mientras que los demás están compuestos por pequeños grupos de usuarios.
- Un grupo está formado por dos cuentas angloparlantes: 'OurWorldInData' y '\_HannahRitchie'. 'OurWorldInData' comparte infografías sobre diversos datos y resultados de investigaciones, mientras que '\_HannahRitchie' es una de las responsables del aspecto investigativo detrás de la creación de estas infografías. En este caso, ambas cuentas publicaron información sobre la desnutrición en Guatemala, lo que hizo que fueran consideradas al realizar el web scraping.
- Otro grupo aislado corresponde a la Federación Nacional de Fútbol de Guatemala, que hizo un llamado oficial a través de una publicación para destacar aspectos importantes relacionados con sus directivos. Al igual que el grupo anterior, este se mantiene aislado debido a su limitada participación en la red fuera de esta publicación.
- Los grupos restantes están conformados por usuarios que publican sobre temas relacionados con el foco principal de la red, pero se encuentran aislados debido a la poca cantidad de conexiones que tienen más allá de sus publicaciones iniciales.
- Todas las métricas de centralidad para los nodos dentro de estos grupos muestran valores extremadamente bajos, lo cual era previsible, ya que se trata de grupos aislados sin interacciones con los nodos del cuerpo principal de la red.

## **(8) Topic & Sentiment Analysis**

## (1) Análisis e identificación de tópicos

Para la identificación de tópicos dentro de los tweets, se utilizará un modelo Latent Dirichlet Allocation (LDA), el cual es el método más común en este ámbito.

```
In [68]: from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

```
In [106...]:
# Vectorization
vectorizer = CountVectorizer(max_df=0.9, min_df=10, stop_words=list(stop_words.union(
    tweet_matrix = vectorizer.fit_transform(df['tweet'])

# Topic Modeling using LDA
lda_model = LatentDirichletAllocation(n_components=10, random_state=random_state)
lda_model.fit(tweet_matrix)
```

```
Out[106...]:
▼      LatentDirichletAllocation
LatentDirichletAllocation(random_state=42)
```

```
In [107...]:
# Function to display the top words per topic
def display_topics(model, feature_names, no_top_words):
    for idx, topic in enumerate(model.components_):
        print(f"\nTopic {idx}:")
        print(", ".join([feature_names[i] for i in topic.argsort()[:-no_top_words - 1]]))
```

```
In [108...]:
# Display topics and their top words
no_top_words = 10 # Number of words to display per topic
feature_names = vectorizer.get_feature_names_out()
display_topics(lda_model, feature_names, no_top_words)
```

Topic 0:  
hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Topic 1:  
presidente, senor, gobierno, mp, guatemala, tiempo, bien, salud, nacional, asi

Topic 2:  
corrupcion, semilla, dia, gobierno, historia, ojala, vaya, presupuesto, viejo, sos

Topic 3:  
guatemala, jajajaja, ustedes, estan, presidente, pueblo, comunidad, diputados, aumento, ley

Topic 4:  
solo, bien, jajaja, sos, mula, hacer, gobiernos, imbecil, nunca, anteriores

Topic 5:  
arevalo, presidente, padre, pais, viva, legado, papa, bernardo, verguenza, usted

Topic 6:  
tambien, ministro, favor, mal, dice, vida, excelente, senor, pueblo, ve

Topic 7:  
asi, solo, periodista, netcenter, pregunta, mierda, puede, ser, tan, periodistas

Topic 8:  
corruptos, mierda, ahora, hace, mp, cuenta, jajajajajaja, pacto, net, anos

Topic 9:  
presidente, gobierno, usted, solo, estan, entonces, idiota, anterior, paso, urge

```
In [109...]: topic_assignments = lda_model.transform(tweet_matrix)
df['topic'] = np.argmax(topic_assignments, axis=1) # Get the topic with the highest probability

# Step 2: Add community information to the DataFrame
df['community'] = df['user_username'].map(partition) # Assuming 'username' is the column with the user's name

# Step 3: Count topic occurrences in each community
topic_counts = df.groupby(['community', 'topic']).size().reset_index(name='count')

# Step 4: Find the most common topic for each community
most_common_topics = topic_counts.loc[topic_counts.groupby('community')['count'].idxmax()]

# Step 5: Display the most used topic and its top words
for _, row in most_common_topics.iterrows():
    topic_number = row['topic']
    top_words = [feature_names[i] for i in lda_model.components_[topic_number].argsort()]
    print(f"\nCommunity {row['community']}: Most Used Topic = {topic_number} (Count: {row['count']})")
    print("Top Words:", ", ".join(top_words))
```

Community 0: Most Used Topic = 0 (Count: 124)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 1: Most Used Topic = 0 (Count: 122)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 2: Most Used Topic = 0 (Count: 68)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 3: Most Used Topic = 0 (Count: 1)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 4: Most Used Topic = 0 (Count: 235)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 5: Most Used Topic = 3 (Count: 1)  
Top Words: guatemala, jajajaja, ustedes, estan, presidente, pueblo, comunidad, diputados, aumento, ley

Community 6: Most Used Topic = 7 (Count: 1)  
Top Words: asi, solo, periodista, netcenter, pregunta, mierda, puede, ser, tan, periodistas

Community 7: Most Used Topic = 5 (Count: 126)  
Top Words: arevalo, presidente, padre, pais, viva, legado, papa, bernardo, verguenza, usted

Community 8: Most Used Topic = 0 (Count: 60)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 9: Most Used Topic = 0 (Count: 107)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 10: Most Used Topic = 0 (Count: 35)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 11: Most Used Topic = 0 (Count: 42)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 12: Most Used Topic = 0 (Count: 26)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 13: Most Used Topic = 0 (Count: 77)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 14: Most Used Topic = 0 (Count: 36)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 15: Most Used Topic = 0 (Count: 4)  
Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 16: Most Used Topic = 2 (Count: 1)  
Top Words: corrupcion, semilla, dia, gobierno, historia, ojala, vaya, presupuesto, viejo, sos

Community 17: Most Used Topic = 0 (Count: 32)

Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 18: Most Used Topic = 0 (Count: 1)

Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 19: Most Used Topic = 0 (Count: 47)

Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

Community 20: Most Used Topic = 9 (Count: 1)

Top Words: presidente, gobierno, usted, solo, estan, entonces, idiota, anterior, pasa, urge

Community 21: Most Used Topic = 0 (Count: 39)

Top Words: hacer, gobierno, va, hijo, puta, creo, mierda, verdad, ver, gran

### Observaciones -->

- En general, todos los tópicos identificados presentan una tonalidad predominantemente negativa, siendo mixta en el mejor de los casos.
- En cuanto a la relación con las comunidades, la negatividad de cada tema sugiere que los temas más utilizados en cada comunidad también tienden a ser negativos.
- Los tópicos más positivos corresponden a los números 1, 3 y 5, que se caracterizan por abordar temas de discusión neutros, con algunas palabras que pueden inferir aspectos positivos, como "viva" y "legado".
- Las comunidades que más utilizan estos tópicos son la 5 y la 7. En cambio, el resto de las comunidades se enfoca principalmente en el tema 0, que abarca temas más negativos o insultos.

## (2) Sentiment Analysis

```
In [89]: from transformers import pipeline
```

```
In [90]: # Load the sentiment analysis pipeline
sentiment_pipeline = pipeline("sentiment-analysis", model="nlptown/bert-base-multilingual-mnli")

# Function to get sentiment
def get_sentiment(tweet):
    result = sentiment_pipeline(tweet)
    return result[0]

# Apply sentiment analysis to the 'tweet' column
df['sentiment'] = df['tweet'].apply(get_sentiment)

# Display the first few rows with sentiment scores
print(df[['tweet', 'sentiment']].head())
```

config.json: 0% | 0.00/953 [00:00<?, ?B/s]

```

pytorch_model.bin:  0% | 0.00/669M [00:00<?, ?B/s]
tokenizer_config.json:  0% | 0.00/39.0 [00:00<?, ?B/s]
vocab.txt:  0% | 0.00/872k [00:00<?, ?B/s]
special_tokens_map.json:  0% | 0.00/112 [00:00<?, ?B/s]
                           tweet \
0 _ confirmado companeres impuesto solo cuenta p...
1 urgente medios faferos informaron ayer acerca ...
2 usaste pegasus espiar detractores obra narcisi...
3 entienden bien estan cuadrando productivareunion
4 presidente vicepresidenta participan sesion so...

sentiment
0 {'label': '1 star', 'score': 0.7896092534065247}
1 {'label': '1 star', 'score': 0.8131591081619263}
2 {'label': '1 star', 'score': 0.3030843138694763}
3 {'label': '4 stars', 'score': 0.404959112405777}
4 {'label': '5 stars', 'score': 0.49949881434440...

```

```
In [96]: df['sentiment'] = df['sentiment'].apply(lambda x: x['score'] if isinstance(x, dict)
```

```

In [100... community_sizes = Counter(partition.values())
largest_communities = community_sizes.most_common(20) # Get the 5 Largest communities

# Step 2: Calculate average sentiment for each community
average_sentiments = {}

for community, _ in largest_communities:
    # Get nodes in the current community
    nodes_in_community = [node for node, comm in partition.items() if comm == community]

    # Get sentiment scores for these nodes
    sentiments = df[df['user_username'].isin(nodes_in_community)]['sentiment']

    # Check if sentiments are numeric and calculate the average
    if not sentiments.empty:
        average_sentiment = sentiments.mean()
        average_sentiments[community] = average_sentiment

# Step 3: Display results
for community, avg_sentiment in average_sentiments.items():
    print(f"Community {community}: Average Sentiment Score = {avg_sentiment:.4f}")

```

Community 4: Average Sentiment Score = 0.5177  
Community 7: Average Sentiment Score = 0.5165  
Community 1: Average Sentiment Score = 0.5163  
Community 0: Average Sentiment Score = 0.5372  
Community 13: Average Sentiment Score = 0.5294  
Community 2: Average Sentiment Score = 0.5121  
Community 9: Average Sentiment Score = 0.5674  
Community 19: Average Sentiment Score = 0.5865  
Community 8: Average Sentiment Score = 0.5625  
Community 17: Average Sentiment Score = 0.5097  
Community 11: Average Sentiment Score = 0.5261  
Community 10: Average Sentiment Score = 0.5442  
Community 14: Average Sentiment Score = 0.5478  
Community 15: Average Sentiment Score = 0.4933  
Community 21: Average Sentiment Score = 0.5514  
Community 12: Average Sentiment Score = 0.5037  
Community 5: Average Sentiment Score = 0.4549  
Community 3: Average Sentiment Score = 0.5036  
Community 20: Average Sentiment Score = 0.4887  
Community 16: Average Sentiment Score = 0.2452

#### **Observaciones -->**

- El sentimiento general de todos los tweets es relativamente neutro, ya que la escala de la puntuación del módulo utilizado emplea 0 como lo más negativo posible y 1 para lo más positivo.
- Las comunidades reflejan este mismo comportamiento, pero existen algunas más positivas que otras.
- La comunidad 19 posee la carga sentimental más positiva, con una puntuación de casi 0.6.
- La comunidad 16 posee una carga sentimental significativamente negativa, con una puntuación de 0.25.
- La discusión entre todas las comunidades y sus tweets implica que existe una percepción mixta del presidente actual, con algunos grupos excesivamente negativos.

## **(9) Results Interpretation**

### **(1) Preguntas Interesantes**

1. **¿Qué aceptación tiene Bernardo Arévalo como presidente de Guatemala actualmente?**

- Bernardo Arévalo cuenta con una aceptación moderada. Existen grupos específicos que son significativamente críticos de él, así como otros con sentimientos mixtos o moderadamente positivos.

## 2. ¿Cuál es la popularidad que tiene en estos momentos?

- En la actualidad, el presidente goza de una popularidad mixta, aunque ligeramente positiva. Hay un alto nivel de interacción y discusión en torno a los tweets relacionados con él, pero las conversaciones más populares dentro del conjunto de datos tienden a centrarse en insultos o críticas hacia su gobierno o ciertas ramas del mismo.

## (2) Impacto de los influencers y las comunidades

Los influencers desempeñan un papel crucial en el flujo de discusiones en la sociedad y en la diseminación de información y opiniones en internet. Un claro ejemplo de esto es el usuario 'UbaldoMacu', quien se dedica a publicar su opinión sobre noticias del país sin una afiliación evidente en su perfil. Este usuario tiene un alto grado de participación en la red, lo que sugiere que sus opiniones podrían influir en otros usuarios y generar más debate sobre ciertos temas.

Otro ejemplo es '\_vaderTG', una cuenta conocida por propagar desinformación y pánico en sus publicaciones. Esta cuenta es altamente controvertida y es probable que algunas de sus publicaciones hayan moldeado la opinión de otros, quienes podrían estar influenciados por información errónea y amarillismo. Sin embargo, es alentador notar que en el conjunto de datos analizado, las cuentas más influyentes y populares pertenecen a entidades oficiales del gobierno, lo que permite concluir que la desinformación no es el factor principal en las discusiones.

Por otro lado, las comunidades formadas en torno a estas discusiones pueden tener efectos imprevistos en la opinión pública. Gracias a los algoritmos de recomendación de publicaciones en las redes sociales, es fácil caer en 'echo chambers' donde prevalece un solo punto de vista. Esto puede llevar a que algunos usuarios tengan una percepción distorsionada del ambiente político actual. Este fenómeno es especialmente preocupante en comunidades que son extremadamente positivas o negativas, como se observa en este análisis, donde hay una comunidad notablemente negativa.

## (3) Conclusiones

- El sentimiento actual sobre Bernardo Arévalo tiende a ser ligeramente positivo, pero el lenguaje y los tópicos utilizados en las discusiones implican ciertos aspectos altamente criticados y percibidos de manera negativa.

- La alta ocurrencia del tópico 0 indica que el público participa activamente en discusiones relacionadas a temas del gobierno y sus funcionarios, pero con un énfasis particular en el presidente y sus acciones.
- La variación en las puntuaciones de sentimiento y la presencia de disidencia en comunidades específicas reflejan que, si bien existe una base de sentimiento positivo, porciones significativas de la población pueden sentirse críticas o insatisfechas.