# Ejemplo de informe

Alvaro allera01@ucm.es
Mario margal15@ucm.es
Mario mcarri17@ucm.es
Diego dielinar@ucm.es

# INTRODUCCION:

Empezamos analizando el número de veces que aparecía cada palabra, sin embargo nos encontramos con que analizaba tambien el número de veces que aparecían signos de puntuación.

Posteriormente arreglamos ese problema como esta comentado en el código y ya obtubimos el número de veces que aparece cada palabra, pero nos encontramos con otro problema y fue que podían aparecer palabras repetidas que solo se diferenciaban por una mayúscula: cyberpunk, Cyberpunk y nos interesaba saber el numero de veces total que aparecía, independientemente de ese factor.

Lo conseguimos arreglar como pone en el codigo, asi que ya teníamos en total el número de veces que aparecía cada palabra, y nos dispusimos con en análisis de dichas palabras e intentar sacar relaciones entre el número de veces que aparecía una palabra y otros factores.

# ANÁLISIS

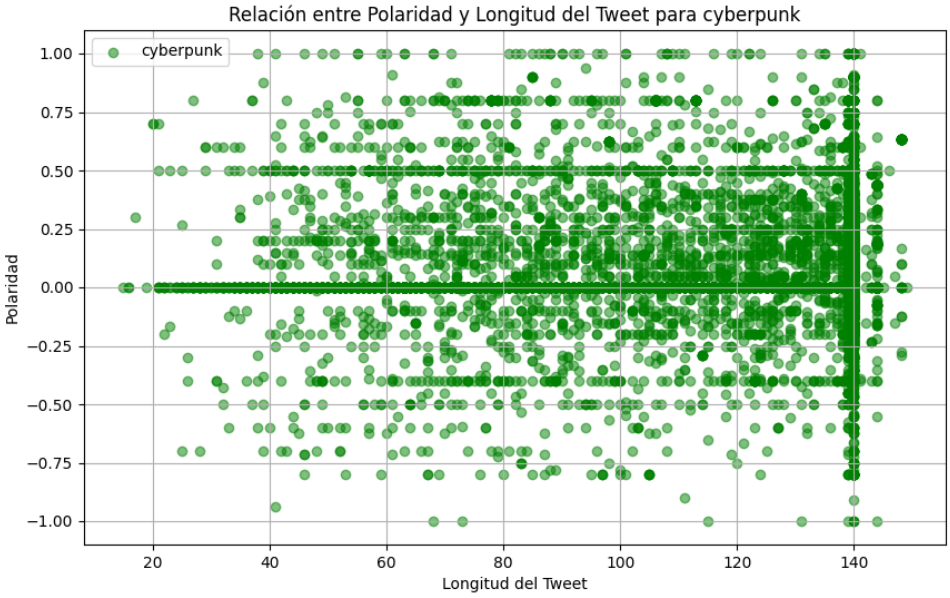
**1.** Analizamos cyberpunk, ya que de eso van los twits, y sacamos un gráfico de polaridad (mostrado como un .png), para relacionar la longitud de los twits con su sentimientos, ya sean positivos o negativos. LLegamos a la conclusión, sorprendentemente, de que había opiniones muy variadas, ya que el gráfico se muestra practicamente neutro.

### Código:

```
def grafico_polaridad_1p(palabra, datos):
    if not os.path.isfile("./grafico_polaridad_" + palabra +".png"):
        df = datos
        df['polarity'] = df['text'].apply(lambda x: TextBlob(x).sentiment.polarity)

        # Calcular la longitud del tweet
        df['tweet_length'] = df['text'].apply(len)

        # Filtrar los tweets que contienen la palabra
        df['contains_palabra'] = df['tokens'].apply(lambda tokens: palabra in tokens)

        # Filtrar solo las filas con la palabra
        df_palabra = df[(df['contains_palabra'])]

        # Crear gráfico de polaridad vs longitud del tweet
        plt.figure(figsize=(10, 6))
```

```
# Graficar tweets con palabra
plt.scatter(df_palabra[df_palabra['contains_palabra']]['tweet_length'],
            df_palabra[df_palabra['contains_palabra']]['polarity'],
            color='green', label=palabra, alpha=0.5)

# Personalización del gráfico
plt.xlabel('Longitud del Tweet')
plt.ylabel('Polaridad')
plt.title('Relación entre Polaridad y Longitud del Tweet para '+palabra)
plt.legend()
plt.grid(True)
plt.savefig("grafico_polaridad_" + palabra +".png")
```

## Resultado:



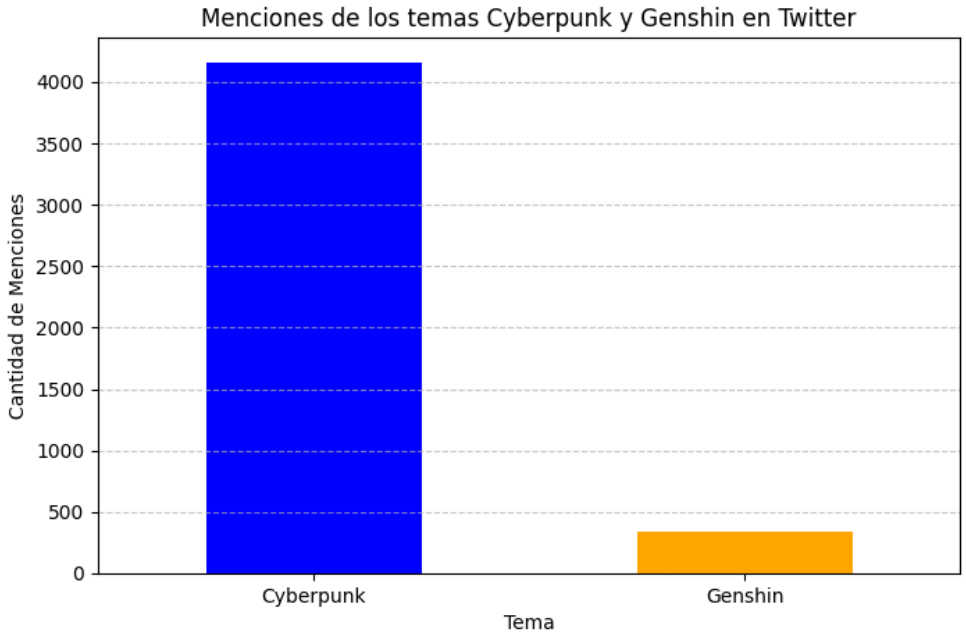

**2.** Analizamos cyberpunk frente al juego genshin impact, viendo que los jugadores de este último lo utilizaban para mencionar su juego.

## Código:

```
def detectar_tema(texto):
    if re.search(r'\b(#?genshinimpact)\b', texto, re.IGNORECASE):
```

```
        return 'Genshin'
    elif re.search(r'\b(cyberpunk\s*2077)\b', texto, re.IGNORECASE):
        return 'Cyberpunk'
    return None

#cantidad de tuits con la palabra cyberpunk y genshin
if not os.path.isfile("./cyberpunkvsgenshin.png"):
    # Aplicar la función a la columna 'text'
    x = df['text'].apply(detectar_tema)

    # Contar la frecuencia de cada tema
    temas_count = x.value_counts()

    plt.figure(figsize=(8, 5))
    temas_count.plot(kind='bar', color=['blue', 'orange'])
    plt.title('Menciones de los temas Cyberpunk y Genshin en Twitter')
    plt.xlabel('Tema')
    plt.ylabel('Cantidad de Menciones')
    plt.xticks(rotation=0)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.savefig("cyberpunkvsgenshin.png")
```

**Resultado:**

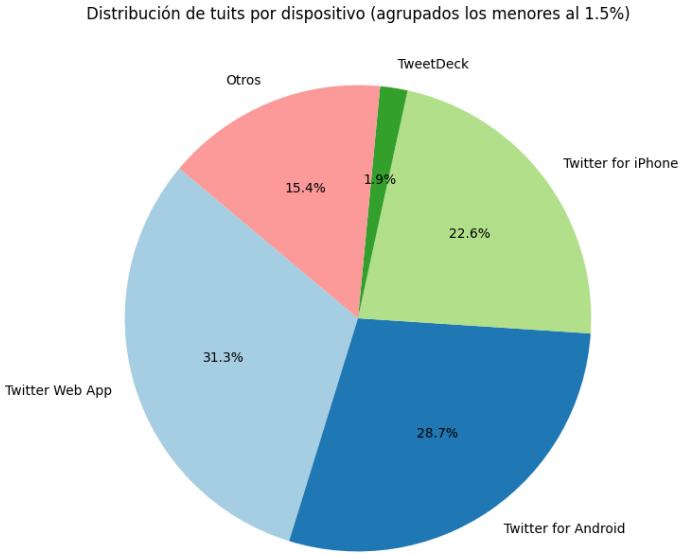


**3.** Analizamos otro aspecto no tan relevante en relación al juego en si, pero si relevante para nosotros como es los dispositivos en los que más se twiteó sobre el tema, habiendo un gran reparto entre la propia aplicación de Twiter, iPhone y Android, siendo la que más en la aplicación de twiter, pero sin sacar gran ventaja a las otras dos. Cabe destacar que este análisis los hicimos con un gráfico circular.

## Código:

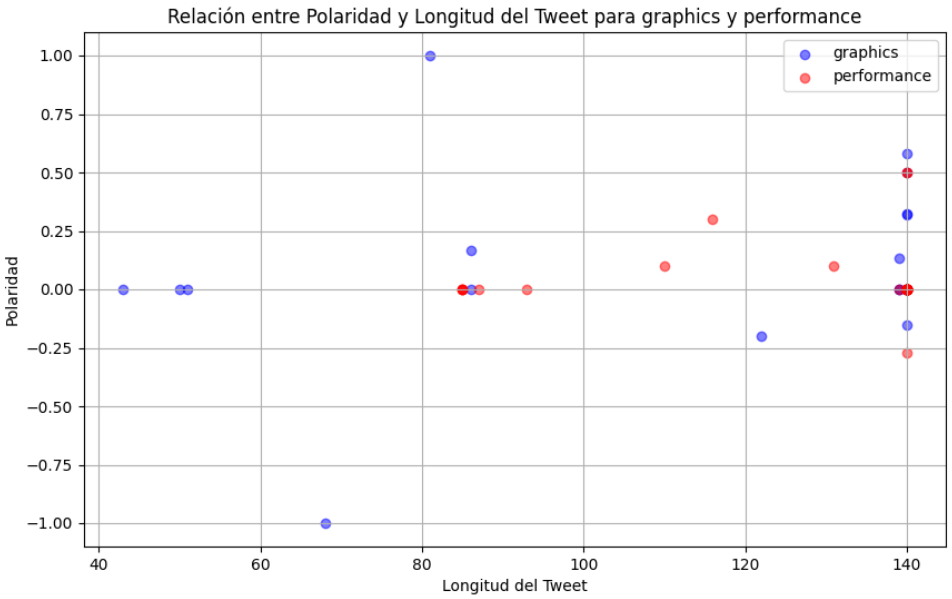
```
def diagrama_circular_dispositivos(df):
    if not os.path.isfile("./dispositivos.png"):
        device_counts = df['source'].value_counts()

        # Calcular el umbral del 1% sobre el total de tuits
        total_tweets = device_counts.sum()
        threshold = total_tweets * 0.015

        # Crear una nueva serie para almacenar los dispositivos agrupados
        device_counts_grouped = device_counts[device_counts >= threshold]
        device_counts_grouped['Otros'] = device_counts[device_counts < threshold].sum()

        # Crear el gráfico circular
        plt.figure(figsize=(8, 8))
        device_counts_grouped.plot(kind='pie', autopct='%1.1f%%', startangle=140, colors=
        plt.title("Distribución de tuits por dispositivo (agrupados los menores al 1.5%)"
        plt.ylabel("")  # Ocultar la etiqueta del eje y
        plt.savefig("dispositivos.png")
```

**Resultado:**



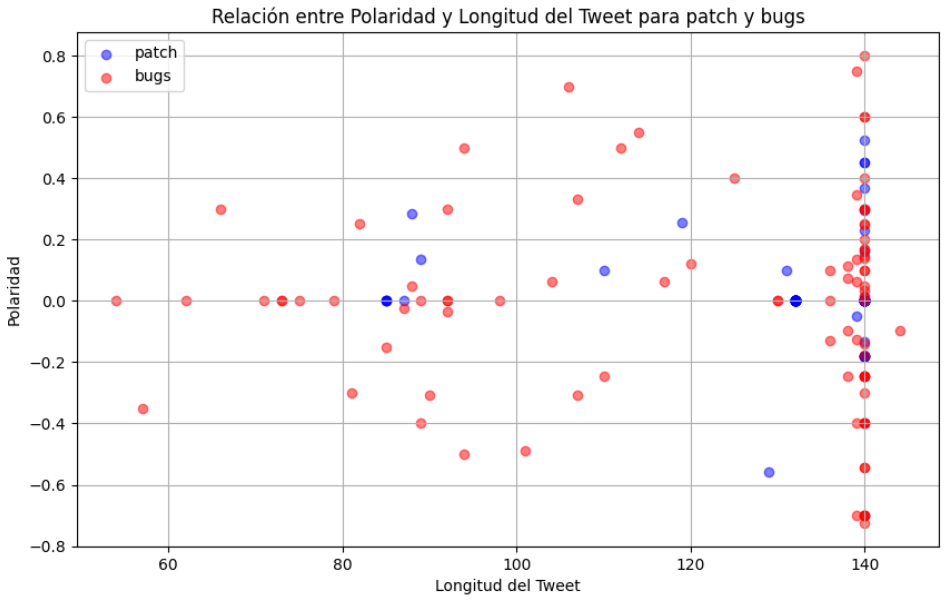

**4.** Con un gráfico de polaridad analizamos los gráficos juntos con la performance del juego, viendo que por parte de ambas hay más sensaciones buenas que malas, llegando a la conclusión de que estos dos aspectos del juego gustaron a los jugadores.

**Resultado:**



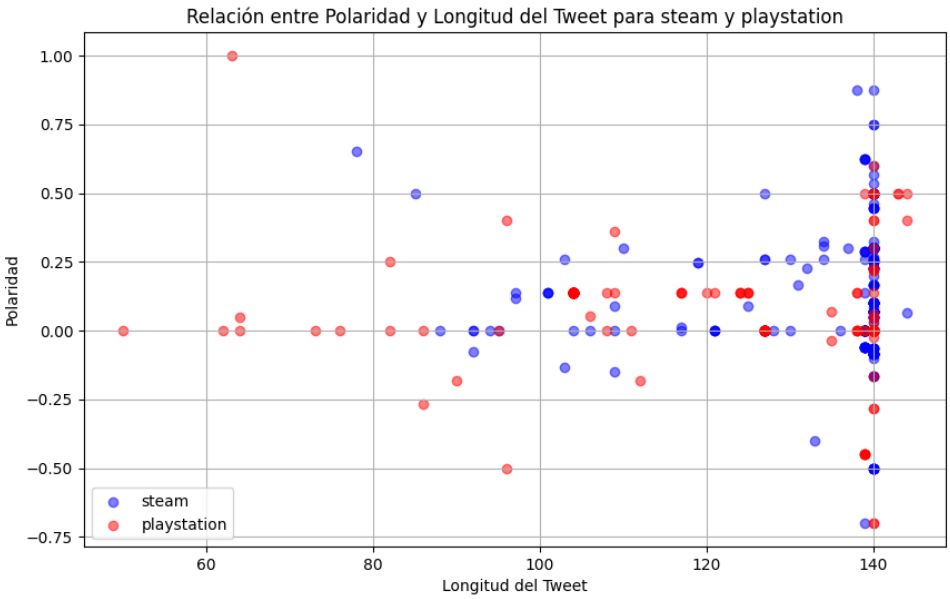

**5.** Analizamos también los bugs junto con los parches del juego para arreglar esos bugs, estando en un gráfico de polaridad bastante igualado, tentando un poco más hacia el positivo, aún así habiendo muchos negativos.

**Resultado:**



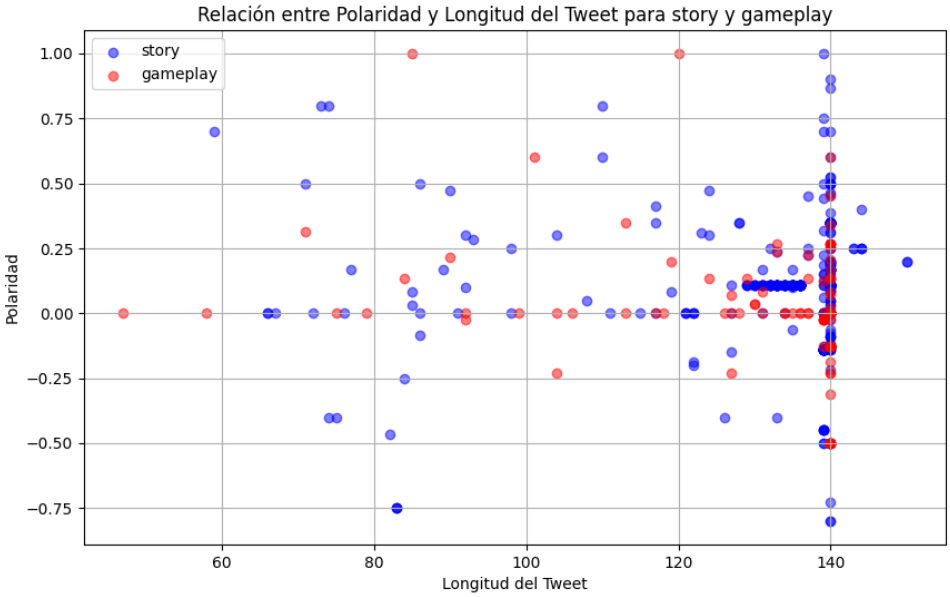

**6.** El juego salió para diferentes plataformas y nos planteamos comparar en un gráfico de polaridad los sentimientos que tuvieron los jugadores de steam frente a los jugadores de playstation y lo primero que llama la atención es que opinaron muchos más jugadores de steam que de playstation, sin embargo para ambos la opinión se mantiene por lo general neutra tirando hacia el lado positivos.

**Resultado:**




**7.** Sacamos otro gráfico de polaridad en relación a la historia y el gameplay, llegando a la conclusión que fueon dos aspectos que también gustaron de salida.

**Resultado:**




## Código del 4, 5, 6 y 7

```
def grafico_polaridad_1p(palabra, datos):
    if not os.path.isfile("./grafico_polaridad_" + palabra +".png"):
        df = datos
        df['polarity'] = df['text'].apply(lambda x: TextBlob(x).sentiment.polarity)

        # Calcular la longitud del tweet
        df['tweet_length'] = df['text'].apply(len)

        # Filtrar los tweets que contienen la palabra
        df['contains_palabra'] = df['tokens'].apply(lambda tokens: palabra in tokens)

        # Filtrar solo las filas con la palabra
        df_palabra = df[(df['contains_palabra'])]

        # Crear gráfico de polaridad vs longitud del tweet
        plt.figure(figsize=(10, 6))

        # Graficar tweets con palabra
        plt.scatter(df_palabra[df_palabra['contains_palabra']]['tweet_length'],
```

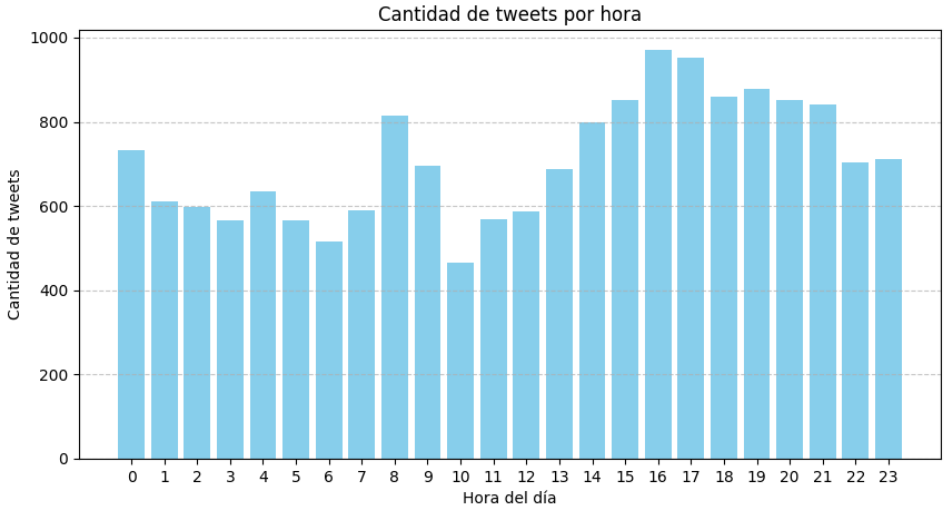
```
                df_palabra[df_palabra['contains_palabra']]['polarity'],
                color='green', label=palabra, alpha=0.5)

    # Personalización del gráfico
    plt.xlabel('Longitud del Tweet')
    plt.ylabel('Polaridad')
    plt.title('Relación entre Polaridad y Longitud del Tweet para '+palabra)
    plt.legend()
    plt.grid(True)
    plt.savefig("grafico_polaridad_" + palabra +".png")
```

**8.** Otra información util para nosotros y no tan relacionada con el juego han sido las horas en las que más se han publicado twits sobre el tema y estas fueron en general más por la tarde sobre todo comprendidas entre las: 16:00-17:00

## Código:

```
def sacar_tuits_horas(bf):
    if not os.path.isfile("./horas.png"):
        # Convierte la columna 'created_at' a tipo datetime
        df['created_at'] = pd.to_datetime(bf['created_at'])

        # Extrae la hora
        df['hora'] = df['created_at'].dt.hour

        # Cuenta la cantidad de tweets por hora
        tweets_por_hora = df['hora'].value_counts().sort_index()

        plt.figure(figsize=(10, 5))
        plt.bar(tweets_por_hora.index, tweets_por_hora.values, color='skyblue')
        plt.xlabel('Hora del día')
        plt.ylabel('Cantidad de tweets')
        plt.title('Cantidad de tweets por hora')
        plt.xticks(tweets_por_hora.index)  # Asegúrate de que todas las horas se muestren
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.savefig("horas.png")
```

## Resultado:

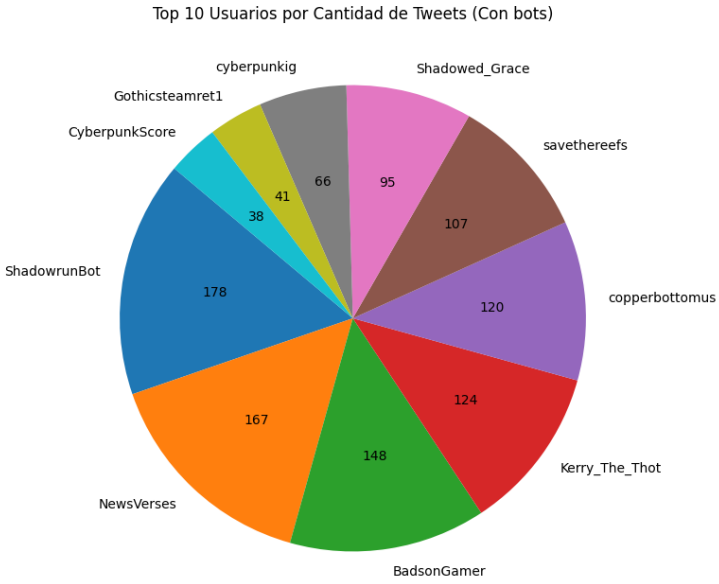


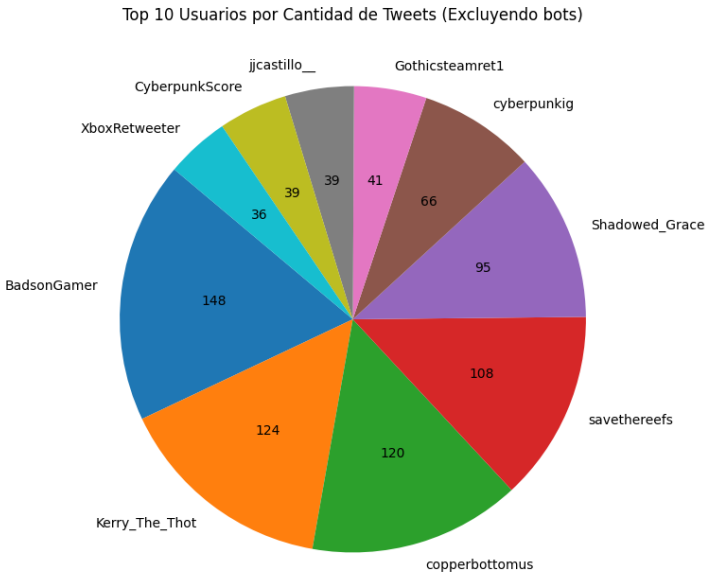

**9.** También hicimos dos gráficos circulares para ver los 10 usuarios que publicaron más twits, primero con bots y luego excluyendo a estos.

## Código:

```
# Obtener los 100 usuarios con más tweets sin bots
if not os.path.isfile("./UsersSinBots.png"):
    filtered_df = df[df['user'] != df['source']]
    user_tweet_counts = filtered_df['user'].value_counts()
    top_users = user_tweet_counts.head(10)
    plt.figure(figsize=(10, 8))
    plt.pie(top_users, labels=top_users.index, autopct=lambda p: f'{int(p * sum(top_users
    plt.title('Top 10 Usuarios por Cantidad de Tweets (Excluyendo bots)')
    plt.savefig('UsersSinBots.png')

# Obtener los 100 usuarios con más tweets con bots
if not os.path.isfile("./UsersBots.png"):
    user_tweet_count = df['user'].value_counts()
    top_user = user_tweet_count.head(10)
    plt.figure(figsize=(10, 8))
    plt.pie(top_user, labels=top_user.index, autopct=lambda p: f'{int(p * sum(top_user) /
    plt.title('Top 10 Usuarios por Cantidad de Tweets (Con bots)')
    plt.savefig('UsersBots.png')
```

## Resultado:


Top 10 Usuarios por Cantidad de Tweets (Con bots)
cyberpunkig
Shadowed_Grace
Gothicsteamret1
CyberpunkScore
savethereefs
66
95
41
38
107
ShadowrunBot
178
120
copperbottomus
124
167
148
Kerry_The_Thot
NewsVerses
BadsonGamer




**10.** Finalmente sacamos los bigramas y trigramas que se repitan un minimo un minimo de 50 veces para ver cuales son las palabras que mas se repiten

## Código:

```
def sacar_bi_tri_gramas(df):
    if not os.path.isfile("./bigramas_trigramas.txt"):
        bigramas = []
        trigramas = []

        for tokens in df['tokens']:
            bigramas.extend(ngrams(tokens, 2))  # Generar bigramas
            trigramas.extend(ngrams(tokens, 3))  # Generar trigramas

        # Contar frecuencias de bigramas y trigramas
        bigramas_frecuentes = Counter(bigramas)
        trigramas_frecuentes = Counter(trigramas)

        with open("bigramas_trigramas.txt", "w") as file:
            file.write("Bigramas más frecuentes:\n")
            for bigrama, frecuencia in bigramas_frecuentes.most_common():
```

```
            if frecuencia > 50:
                file.write(f"{bigrama}: {frecuencia}\n")

        file.write("\nTrigramas más frecuentes:\n")
        for trigrama, frecuencia in trigramas_frecuentes.most_common():
            if frecuencia > 50:
                file.write(f"{trigrama}: {frecuencia}\n")

# Mostrar los 10 bigramas y trigramas más comunes
#print("Bigramas más comunes:", bigramas_frecuentes.most_common(20))
#print("Trigramas más comunes:", trigramas_frecuentes.most_common(20))
```

### Resultado:

Un ejemplo de **los 10 Bigramas** más utilizados son:

(‘cyberpunk’, ‘2077’): 3515

(‘this’, ‘cyberpunk’): 1000

(‘is’, ‘this’): 972

(‘rt’, ‘cyberpunkcortes’): 922

(‘cyberpunkcortes’, ‘is’): 922

(’ ‘, ’s’): 642

(‘i’, “’m”): 636

(‘in’, ‘the’): 541

(‘a’, ‘cyberpunk’): 516

(‘it’, “’s”): 470

Un ejemplo de **los 10 Trigramas** más utilizados son:

(‘is’, ‘this’, ‘cyberpunk’): 937

(‘rt’, ‘cyberpunkcortes’, ‘is’): 922

(‘cyberpunkcortes’, ‘is’, ‘this’): 922

(‘boys’, ‘being’, ‘boys’): 336

(‘being’, ‘boys’, ‘but’): 336

(‘boys’, ‘but’, ‘cyberpunk’): 336

(‘but’, ‘cyberpunk’, ‘genshinimpact’): 336

(‘xiao’, ‘aether’, ‘xiaoaether’): 336

(‘rt’, ‘farevalee9s’, ‘boys’): 335

(‘farevalee9s’, ‘boys’, ‘being’): 335

(‘cyberpunk’, ‘2077’, ‘and’): 324