BACARO TECH

Software Solutions

Twitter Data Analysis

Manual Técnico

Manejo de Datos no Estructurados

Dr. Francisco J. Moo Mena

Integrantes:
Daniel Fernando Baas Colonia
Rodrigo Castilla López
Luis Gerardo Castillo Pinkus
Rafael Rodríguez Guzmán

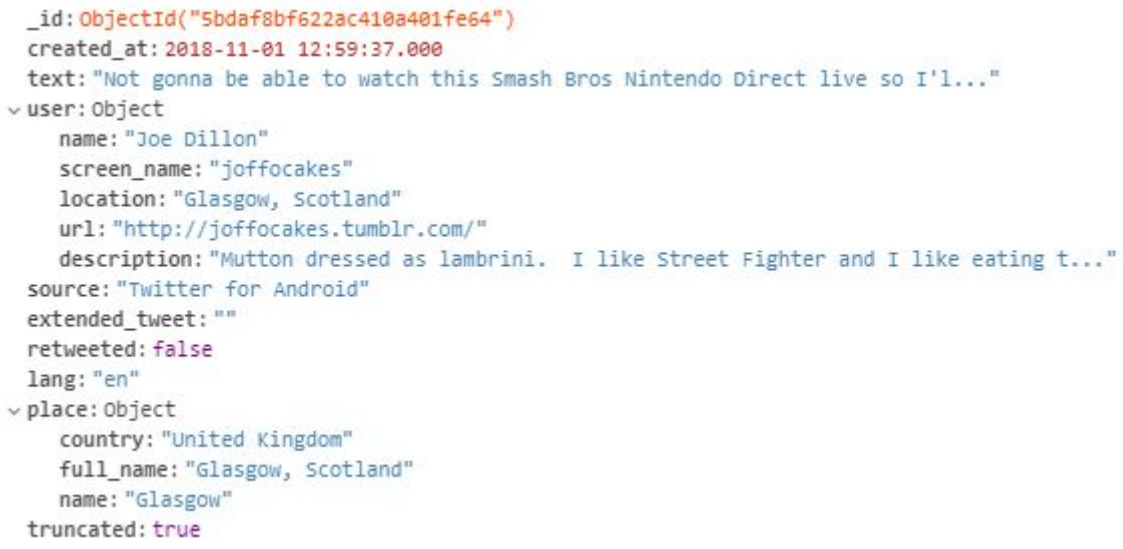
Instalaciones necesarias

- Python: el proyecto fue realizado con la versión 3.6 de Python
    - https://www.python.org/downloads/release/python-367/
- pip3: se usó el manejador de paquetes pip3 para instalar las bibliotecas de Python necesarias.
    - sudo apt install python3-pip
- Tweepy: se usó la biblioteca de tweepy para realizar la conexión a la api de Twitter.
    - pip3 install tweepy
- Matplotlib: biblioteca que permite generar gráficas.
    - pip3 install matplotlib
- Pymongo: driver para Python usado para manejar la interacción con la base de datos de MongoDB.
    - pip3 install pymongo
- WordCloud: biblioteca que permite generar nubes de palabras con Python.
    - pip3 install wordcloud
- Numpy: biblioteca que permite realizar cómputo científico con Python.
    - pip3 install numpy
- MongoDB: base de datos no relacional. Se usa para almacenar la información de los tweets recolectados.
    - https://www.mongodb.com/cloud/atlas

MongoDB como base de datos

Esta aplicación hace uso de MongoDB como base de datos no estructurada. Más específicamente, se usa el clúster de Atlas en su versión gratuita.

Cada tweet representa un documento, y la información se almacena es la siguiente:



Cabe destacar que no todos los tweets contienen la información completa para todos los campos. Por ejemplo, sólo 361 de los 72595 tweets totales contienen información en el campo *place.*

Documentación del código

En esta aplicación está compuesta de dos partes: la recolección de tweets y el análisis de ellos. La colección se realiza ejecutando el script *get_tweets.py*, mientras que el análisis de los datos se realiza ejecutando el script *procesing.py*.

procesing.py

Lo primero que hace al ejecutar este archivo es crear una conexión al servidor Atlas de MongoDB. En este servidor se encuentra la base de datos que contiene todos los tweets. Una vez encontrada la base de datos y la colección, se muestra en pantalla un mensaje al usuario indicando la conexión exitosa. Algo importante es que client = MongoClient() establece una conexión con MongoDB. Dado que usamos el clúster de Atlas de MongoDB, esa función lleva una cadena como parámetro. Dicha cadena se tiene que generar desde la página de Atlas una vez hayas creado tu servidor.

```
15  # Creamos una conexión con el servidor
16  client = MongoClient(
17      "mongodb://rafael:S7BNKu4WAj2z3ca6@nonstructureddatamanagementclass-shard-00-00-xa2lb.mongodb.net:27017,"
18      "nonstructureddatamanagementclass-shard-00-01-xa2lb.mongodb.net:27017,"
19      "nonstructureddatamanagementclass-shard-00-02-xa2lb.mongodb.net:27017/twitter_db?ssl=true&replicaSet"
20      "=NonStructuredDataManagementClass-shard-0&authSource=admin&retryWrites=true")
21
22  # Nuestra base de datos se llamará "nintendo"
23  db_name = "nintendo"
24
25  # Intentamos realizar una conexión con la base de datos. Si la BD no existe, la crea.
26  db = client[db_name]
27
28  # Recuperamos el nombre de todas las bases de datos en el servidor
29  db_list = client.list_database_names()
30
31  # Revisamos si la base de datos existe
32  if db_name in db_list:
33      print("\nLa base de datos \"" + db_name + "\" existe!")
34
35  # Una colección es un grupo de documentos almacenados en MongoDB
36  # Nuestra colección se llamara "tweets"
37  collection_name = "tweets"
38
39  # Crea una nueva colección o recupera el contenido de una existente
40  # Si la colección se está creando, MongoDB no la va a crear hasta que se le inserte documentos
41  collection = db[collection_name]
42
43  # Recuperamos el nombre de todas las colecciones
44  collection_list = db.list_collection_names()
45
46  # Revisamos si la colección existe. Si la colección creada no tiene elementos, MongoDB no la va a crear hasta
47  # insertar un documento
48  if collection_name in collection_list:
49      print("La coleccion \"" + collection_name + "\" existe!\n")
```

is_valid_tweet(tweet): recibe una cadena de texto que contiene el tweet y revisa si contiene alguna palabra correspondiente a una lista de términos que no son de nuestro interés. Si la cadena contiene alguna palabra se retorna falso, sino, se retorna verdadero, indicando que el tweet si es de nuestro interés.

```python
# Verifica que el tweet no sea uno relacionado con términos que no son relevantes para nosotros
def is_valid_tweet(tweet):
    # Dados los tags usados para recopilar los tweets, no todos resultaron tener información útil. Así que se tiene
    # una lista de tweets cuyo tema no es de nuestro interés
    filter_list = [
        "64",
        "classic",
        "n64",
        "#lookingfor",
        "$239.99",
        "#splatoon2",
        "wii"
    ]

    for fw in filter_list:
        if fw in tweet:
            return False

    return True

```

generate_word_cloud(tweets): esta función crea una imagen de una nube de palabras. Primero recibe una lista de tweets. El contenido de texto de cada tweet se concatena en una variable para poder pasar la cadena de texto completa a la biblioteca de wordcloud. Una vez que todos los textos de los tweets están unidos generamos una lista de palabras que el wordcloud va ignorar y no las va a colocar al final. Después que se genero la lista se carga una imagen con una forma que tomará la nube de palabras conocida como máscara (mask). En nuestro la máscara tiene forma de Mario (Super Mario Bros) pues va con la temática de Nintendo. Finalmente se usa la función WordCloud para generar la nube de palabras y la guarda en una imagen en la ruta */plots/wordcloud.png*

```python
# Genera una imagen .png con una nube de palabras de los términos más usados en los tweets
def generate_wordcloud(tweets):
    print("\nGenerando nube de palabras...\n")

    tweet_list = []
    for tweet in tweets:
        tweet_list.append(tweet["text"])

    # Unimos el texto de todos los tweets en una sola variable
    text = "\n".join(tweet_list)

    # Cargamos la lista de stopwords y añadimos más que consideramos no relevantes para mostrar en la nube de palabras
    # Las stopwords son palabras que serán ignoradas
    stopwords = set(STOPWORDS)
    stopwords.add("nintendodirect")
    stopwords.add("smash")
    stopwords.add("bros")
    stopwords.add("smashbros")
    stopwords.add("supersmashbros")
    stopwords.add("ultimate")
    stopwords.add("smashbrosultimate")
    stopwords.add("supersmashbrosultimate")
    stopwords.add("co")
    stopwords.add("nintendo")

    # Cargamos la máscara que usaremos para darle forma a la nube
    mask = np.array(Image.open("sources/mario.png"))

    # Genera una nube de palabras. Las stopwords serán ignoradas y no apareceran en la gráfica
    wordcloud = WordCloud(width=1079, height=1623, max_words=10000, relative_scaling=1, stopwords=stopwords,
                          mask=mask, contour_color="white").generate(text)

    # Guardamos la nube de palabras generada en una imagen en la misma carpeta del script
    wordcloud.to_file("plots/wordcloud.png")
```

generate_app_plot(tweets): esta función genera una gráfica de pastel que muestra las siete aplicaciones más usadas para realizar los tweets. Recibe como parámetro una lista de tweets, y genera una lista de objetos de tipo {“device”: aplicación, “count”: número de tweets para esa aplicación}, para ir contabilizando cuántos tweets se realizaron desde cada aplicación. Dada la enorme cantidad de aplicaciones, sólo se seleccionaron las siete más usadas. Al finalizar, se genera la gráfica de pastel y la guarda en una imagen en la ruta */plots/applications.png*.

```python
# Genera una gráfica de pastel con las aplicacciones desde donde se realizaron la mayoría de los tweets
def generate_app_plot(tweets):
    print("\nGenerando gráfica de dispositivos...\n")

    # Obtenemos una lista de objetos con el nombre de la aplicación y el número de tweets
    source_list = generate_source_list(tweets)

    # Dado que hay una enorme cantidad de aplicaciones, sólo vamos a mostrar las 7 más usadas, y las demás se contarán
    # dentro de "Otras" aplicaciones

    src_count = 0
    max_src = 7
    src_plot = []

    # Revisamos la lista de aplicaciones y generamos una con las más usadas para hacer los tweets
    for src in source_list:
        if src_count < max_src:
            src_plot.append(src)
            src_count = src_count + 1

        if src_count == max_src:
            src_plot.append({
                "source": "Others",
                "count": 1
            })

            src_count = src_count + 1

        if src_count > max_src:
            src_plot[max_src]["count"] = src_plot[max_src]["count"] + 1

    # Nombre de las aplicaciones
    labels = []
    # Número de tweets por aplicación
    sizes = []
```

```python
    # Añadimos la información para las etiquetas en la gráfica
    for src in src_plot:
        labels.append(src["source"])
        sizes.append(src["count"])


    # Ajusta el tamaño de la gráfica para que se adapte al tamaño del contenido
    plt.tight_layout()
    # Ajustamos el tamaño de la letra
    plt.rcParams["font.size"] = 7.0
    # Añadimos un título
    plt.title("Most used applications")
    # Generamos una gráfica de pastel
    plt.pie(sizes, labels=labels, shadow=True, radius=10, autopct="%0.2f%%")
    plt.axis("equal")

    # Guardamos la gráfica en una imagen en la misma carpeta que este script
    plt.savefig("plots/applications.png", dpi=300)

    plt.show(block=False)
    plt.close()
```

generate_country_plot(tweets): esta función genera una gráfica de barras que muestra los diez países que realizaron más tweets. Recibe como parámetro una lista de tweets, y genera una lista de objetos de tipo {“country”: país, “count”: número de tweets para ese país}, para ir contabilizando cuántos tweets se realizaron desde cada cada país. Dada la enorme cantidad de países, sólo se seleccionaron diez que más tweets realizaron. Al finalizar, se genera la gráfica de barras y la guarda en una imagen en la ruta */plots/countries.png*. Como nota se puede decir que, como no todos los tweets contienen información sobre localización, la gráfica representa alrededor de 300-400 tweets, que son los que si contienen información de localización.

```
204  # Genera una gráfica de barras con los países que más tweets realizaron.
205  # Dado que no todos los tweets tienen información de ubicación, esta lista de tweets tiene alrededor de 300+ tweets
206  def generate_country_plot(tweets):
207      print("\nGenerando gráfica de países...\n")
208
209      # Obtenemos una lista de objetos que contiene el nombre del país con el número de tweets
210      country_list = generate_country_list(tweets)
211
212      # Dado que la lista de países es bastante grande, sólo mostraremos los primeros 10 con más tweets
213
214      country_count = 0
215      max_country = 10
216      country_chart = []
217
218      # Generamos una lista con los 10 países que más tweetearon
219      for country in country_list:
220          if country_count < max_country:
221              country_chart.append(country)
222              country_count = country_count + 1
223
224      # Nombre de los países
225      labels = []
226      # Número de tweets
227      sizes = []
228
229      # Añadimos la información a las etiquetas de la gráfica de barras
230      for country in country_chart:
231          labels.append(country["country"])
232          sizes.append(country["count"])
233
234      # Prepara la información necesaria para generar la gráfica de barras
235      y_pos = np.arange(len(labels))
236      plt.yticks(y_pos, labels)
```

```
    # Añadimos una etiqueta al eje vertical
    plt.ylabel("Country")
    # Añadimos una etiqueta al eje horizontal
    plt.xlabel("# of tweets")
    # Añadimos un título
    plt.title("Top 10 countries with the most tweets")
    # Ajusta el tamaño de la gráfica para que se adapte al tamaño del contenido
    plt.tight_layout()

    # Añade una etiqueta que muestra el número de tweets a lada de cada barra
    for i, v in enumerate(sizes):
        plt.text(v + 5, i - 0.1, str(v), color="#1F76B3", fontweight="bold")

    # Genera la gráfica
    plt.barh(y_pos, sizes)

    # Guarda la gráfica como imagen en la misma carpeta que este script
    plt.savefig("plots/countries.png", dpi=300)

    plt.show(block=False)
    plt.close()
```

generate_language_plot(tweets): esta función genera una gráfica de pastel que muestra los siete lenguajes más usados para realizar los tweets. Recibe como parámetro una lista de tweets, y genera una lista de objetos de tipo {“lang”: lenguaje, “count”: número de tweets para ese lenguaje}, para ir contabilizando cuántos tweets se realizaron por cada lenguaje. Dada la enorme cantidad de lenguajes, sólo se seleccionaron los siete más usados. Al finalizar, se genera la gráfica de pastel y se guarda en una imagen en la ruta */plots/languages.png*.

```python
# Genera una gráfica de pastel que representa los idiomas más usados en los tweets
# Dado que hay muchos idiomas solo te van a tomar como máximo los definidos en la variable max_lang
def generate_language_plot(tweets):
    print("\nGenerando gráfica de idiomas...\n")

    # Obtenemos una lista de objetos con las siglas del idioma y el número de tweets en ese idioma
    language_list = generate_language_list(tweets)

    # Dado que hay una enorme cantidad de idiomas, sólo tomaremos un número limitado, definido por la variable max_lang
    lang_count = 0
    max_lang = 7
    lang_plot = []

    # Revisamos la lista de aplicaciones y generamos una con las más usadas para hacer los tweets
    for lang in language_list:
        if lang_count < max_lang:
            lang_plot.append(lang)
            lang_count = lang_count + 1

        if lang_count == max_lang:
            lang_plot.append({
                "language": "Others",
                "count": 1
            })

            lang_count = lang_count + 1

        if lang_count > max_lang:
            lang_plot[max_lang]["count"] = lang_plot[max_lang]["count"] + 1

    # Listado de idiomas a usar como etiquetas en la gráfica
    labels = []
    # Número de tweets por idioma
    sizes = []
```

```python
    # Añadimos la información para las etiquetas en la gráfica
    for lang in lang_plot:
        labels.append(lang["language"])
        sizes.append(lang["count"])

    # Ajusta el tamaño de la gráfica para que se adapte al tamaño del contenido
    plt.tight_layout()
    # Ajustamos el tamaño de la letra
    plt.rcParams["font.size"] = 7.0
    # Añadimos un título
    plt.title("Most used languages")
    # Generamos una gráfica de pastel
    plt.pie(sizes, labels=labels, shadow=True, radius=10, autopct="%0.2f%%")
    plt.axis("equal")

    # Guardamos la gráfica en una imagen en la misma carpeta que este script
    plt.savefig("plots/languages.png", dpi=300)

    plt.show(block=False)
    plt.close()
```

print(tweet)_single_tweet y print_tweets(tweets): la primera función imprime en pantalla el contenido completo de un solo tweet, mientras que la segunda itera sobre la lista completa de tweets, llamando a print_single_tweet(tweet) para la impresión en pantalla por cada elemento.

```python
# Imprime el contenido del tweet recuperado de MongoDB
def print_single_tweet(tweet):
    print("\ncreated_at: " + str(tweet["created_at"]))
    print("text: " + tweet["text"])
    print("user.name: " + tweet["user"]["name"])
    print("user.screen_name: " + tweet["user"]["screen_name"])
    print("user.location: " + str(tweet["user"]["location"]))
    print("user.url: " + str(tweet["user"]["url"]))
    print("user.description: " + str(tweet["user"]["description"]))
    print("source: " + tweet["source"])
    print("extended_tweet: " + tweet["extended_tweet"])
    print("retweeted: " + str(tweet["retweeted"]))
    print("lang: " + tweet["lang"])
    print("place.country: " + tweet["place"]["country"])
    print("place.full_name: " + tweet["place"]["full_name"])
    print("place.name: " + tweet["place"]["name"])
    print("truncated: " + str(tweet["truncated"]))


# Imprimimos todos los tweets de un conjunto dado
def print_tweets(tweets):
    for tweet in tweets:
        print_single_tweet(tweet)
```

generate_plot_list.py

Este script permite generar listas de objetos dados ciertos campos de los tweets. Se usa para generar la lista de aplicaciones, la de países y la de lenguajes. Estas listas tiene una forma:

{“field”: campo que se utiliza para contabilizar, “count”: número de ocurrencias}

“field” puede tomar el valor de “device”, “country” o “lang”, dependiendo de si la lista a generar es una de aplicaciones, países o lenguajes. “count” siempre representa el número de tweets que se han contabilizado para dicho campo de referencia.

Al final de la ejecución, retorna una lista de objetos basados en la estructura mencionada anteriormente y el campo sobre el cuál se contabilizaron los tweets.

is_in_list(element, field): verifica si el elemento ya se encuentra en la lista.

add_element(element, field): añade un nuevo elemento a la lista.

```python
# Lista de todas las aplicaciones desde las que se realizaron los tweets con sus conteos
element_list = []

# Verifica si la aplicación ya se está contabilizando o es una nueva
def is_in_list(element, field):
    # La lista está vacía
    if len(element_list) == 0:
        return False

    # Verifica que la aplicación se encuentre en la lista
    for elmt in element_list:
        if element == elmt[field]:
            return True

    return False
```

```python
# Añadimos una nueva aplicación a la lista de fuentes de tweets
def add_element(element, field):
    new_element = {
        field: element,
        "count": 1
    }

    element_list.append(new_element)


# Imprime la lista de dispositivos con la cantidad de tweets de cada uno
def print_elements(elmt_list, field):
    for elmt in elmt_list:
        print(field + ": " + elmt[field])
        print("total: " + str(elmt["count"]))
```

generate_source_list(tweets): genera una lista que contabiliza el número de tweets por cada aplicación bajo la cual fueron realizados.

```python
# Organiza las aplicaciones desde las que se realizaron los tweets, aumentando el contendo de tweet por aplicación o
# añadiendo nuevas aplicaciones al conteo
def generate_source_list(tweets):
    element_list.clear()

    field = "source"

    for tweet in tweets:
        if is_in_list(tweet[field], field):
            increment_element(tweet[field], field)
        else:
            add_element(tweet[field], field)

    new_list = sort_list(element_list, "count")

    #print_elements(new_list, field)

    return new_list
```

generate_country_list(tweets): genera una lista que contabiliza el número de tweets por cada país del cual se realizó el tweet.

```
75  def generate_country_list(tweets):
76      element_list.clear()
77
78      field = "country"
79
80      for tweet in tweets:
81          if is_in_list(tweet["place"]["country"], field):
82              increment_element(tweet["place"]["country"], field)
83          else:
84              add_element(tweet["place"]["country"], field)
85
86      new_list = sort_list(element_list, "count")
87
88      #print_elements(new_list, field)
89
90      return new_list
```

generate_language_list(tweets): genera una lista que contabiliza el número de tweets por cada lenguaje en el que se realizó cada tweet.

```
92  def generate_language_list(tweets):
93      element_list.clear()
94
95      field = "language"
96
97      for tweet in tweets:
98          if is_in_list(tweet["lang"], field):
99              increment_element(tweet["lang"], field)
100         else:
101             add_element(tweet["lang"], field)
102
103     new_list = sort_list(element_list, "count")
104
105     #print_elements(new_list, field)
106
107     return new_list
```

get_tweets.py

Este script realiza una conexión al servidor para acceder a la base de datos, y de no existir, crea una. Una vez que ya está conectado a la base de datos accede

a la colección donde se almacenarán los tweets que se recuperen. El siguiente paso es realizar una conexión a la api de Twitter para empezar un stream de tweets que se están publicando en este mismo instante. Los tweets que se encontrarán corresponden a una serie de términos clave proporcionados. El stream permanecerá activo mientras el script esté en ejecución y haya conexión al clúster de Atlas. Para terminar su ejecución es necesario acabar con el proceso de manera manual.

MyStreamListener(tweepy.StreamListener): es una clase que permite el stream de tweets. Dentro de ella obtenemos cada tweet por el objeto "*self*". Debemos acceder a los campos que sean de nuestro interés del tweets, y almacenarlos en una variable para poder guardar dicha información en MongoDB. *collection_insert_one(tweet)* inserta el tweet que generamos a la base de datos.

```
# Sirve para realizar el stream de tweets
class MyStreamListener(tweepy.StreamListener):

    # Para cada tweets obtenido, recuperamos la información que se muestra
    def on_status(self, status):
        tweet = {
            "created_at": status.created_at,
            "text": status.text,
            "user": {
                "name": status.user.name,
                "screen_name": status.user.screen_name,
                "location": status.user.location,
                "url": status.user.url,
                "description": status.user.description
            },
            "source": status.source,
            "extended_tweet": "",
            "retweeted": status.retweeted,
            "lang": status.lang,
            "place": {
                "country": "",
                "full_name": "",
                "name": ""
            },
            "truncated": status.truncated
        }
```

```
        # Ciertos atributos de los tweets pueden o no pueden estar, así que debemos verificar que si se encuentren
        # antes de acceder a ellos u obtendremos errores por null
        if hasattr(status, "extended_tweet"):
            if (hasattr(status.extended_tweet, "full_text")):
                tweet["extended_tweet"] = status.extended_tweet.full_text
        if hasattr(status, "place"):
            if hasattr(status.place, "country"):
                tweet["place"]["country"] = status.place.country
            if hasattr(status.place, "full_name"):
                tweet["place"]["full_name"] = status.place.full_name
            if hasattr(status.place, "name"):
                tweet["place"]["name"] = status.place.name

        # Imprimimos los tweets para verificar que el stream sigue activo
        print(status.text + "\n")

        # Insertamos el tweet a la colección
        inserted = collection.insert_one(tweet)
```

consumer_key, consumer_secret, access_token y *access_token_secret* es información proporciona por Twitter. Para obtener dichas clases es necesario registrarse en Twitter como desarrollador (https://developer.twitter.com/).

Una vez obtenidas las llaves, es necesario realizar una autenticación con ellas e iniciar el stream de tweets. Se establecieron términos que filtran qué tweets son los que queremos obtener.

```
# Datos de acceso a la api
consumer_key = '0zLVRnLCdP0JuUxjsMlB5P6id'
consumer_secret = 'UEOvI6l0QFgaWxIzee3LJMOt5C59M5YhL9yh40dOq0ZGBXs85n'
access_token = '940271662959906816-CEzHkOji1IYR1Z1VZeXdSYDrNetgRVT'
access_token_secret = 'wXev8vS56LhX06adGOgDX8HFTrAcjhwSztQGBGlzQePVA'

# Autenticación y validación con la api
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

# Iniciamos un nuevo stream de tweets
myStreamListener = MyStreamListener()
myStream = tweepy.Stream(auth=api.auth, listener=myStreamListener, tweet_mode='extended')

# Establecemos los temas sobre los que queremos obtener información
myStream.filter(
    track=['Nintendo', 'NintendoDirect', 'NintendoSwitch', 'SmashBros', 'SuperSmashBros', 'SuperSmashBrosUltimate',
           'SmashBrothers', 'SuperSmashBrothers', 'SuperSmashBrothersUltimate', 'SmashUltimate'])
```