

Práctica 2: Limpieza y análisis de los datos

Enrique Vilanova Vidal* Brais Suarez Souto

5 de enero de 2021

*Todos los que han colaborado en el proyecto

Notas previas al ejercicio:

Hemos contestado a las preguntas de acuerdo a lo indicado al ejercicio, sin embargo debido a la extensión del trabajo y del código, únicamente algunas partes del código y su desarrollo se han incluido en este texto. El objetivo es facilitar la comprensión del desarrollo de las diversas partes del trabajo. El código en su totalidad se puede encontrar en el enlace que se facilita a GitHub.

El código se ha hecho sobre *Google Colab* para facilitar la reproducción y ejecución del trabajo. La primera parte del código está implementado en *Python* y la segunda parte en *R*. Las dos partes se han implementado sobre el mismo notebook.

En *Python* se ha implementado la limpieza general de datos, el desarrollo de métricas, transformaciones exponenciales de los datos, gráficas de correlación y todo lo relacionado con *Natural Language Processing*.

En *R* se ha implementado modelos de regresión, análisis de componente principal y test estadísticos.

Con el fin de aumentar la reproducibilidad del trabajo, en el repositorio se incluyen objetos pickle y archivos csv con los datasets intermedios.

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

1.1. Descripción del dataset.

El conjunto de datos objeto de análisis se ha obtenido a partir de este enlace en Kaggle y esta constituido por 10 datasets (uno por cada mercado regional: Canada, Alemania, Francia, Gran Bretaña, India, Japon, Korea, Méjico, Rusia, Estados Unidos) de 16 características (columnas) que representan los videos que han llegado a ser Trending en sus respectivas regiones.

Las características representadas son las siguientes:

- **video_id**: ID único para cada video.
- **trending_date**: fecha en la que el video entró en Trend.
- **title**: Título del video.
- **channel_title**: Título del canal al que se ha subido el vídeo.
- **category_id**: ID de la categoría del video. (El nombre de cada categoría viene dado en un fichero json externo)
- **publish_time**: Día y hora a la que se ha subido el video.
- **tags**: SEO tags introducidas en el video.
- **view**: Número de visualizaciones que ha tenido el video.
- **likes**: Número de likes.
- **dislikes**: Número de dislikes.
- **comment_count**: Número de comentarios.
- **thumbnail_link**: Link dónde se almacena la imagen de la thumbnail.
- **comments_disabled**: Booleano que define si se han desactivado los comentarios.
- **ratings_disabled**: Booleano que define si se han desactivado los ratings.

- **video_error_or_removed**: Booleano que define si se han el video se ha borrado por algún error o manualmente.
- **description**: Descripción del video.

1.2. ¿Por qué es importante y qué pregunta/problema pretende responder?

El dataset se ha elegido como continuación o extensión de los datos recopilados en la práctica 1. A modo de recordatorio, en la práctica anterior nuestro objetivo era reconocer los temas que resultaban de mayor interés a la mayoría de la población en un momento en particular. Para cumplir este objetivo nos concentramos en la extracción de datos de las plataformas: *Meneame*, *Reddit* y *Twitter*. Sin embargo otra plataforma muy popular es *Youtube* de la que podemos extraer similar información, además los atributos que obtenemos de la plataforma *Youtube* son comparables a los que obtenemos de las tres plataformas que estudiamos en la primera práctica y un procesado de datos muy similar se puede establecer para los cuatro plataformas.

A partir del conjunto de datos para *Youtube* se plantean diferentes preguntas y análisis que pueden que pueden extrapolarse fácilmente a los datos presentados en la primera práctica. En nuestro caso hemos intentado responder a varias preguntas:

- ¿Qué categorías son mas populares en cada mercado?
- ¿Cuánto tarda un video en hacerse Trending?
- ¿Cuales son las palabras más utilizadas?
- ¿Cuales son las tags más relevantes?
- ¿Cuales son las temas más importantes?
- ¿Existen diferencias en el vocabulario que se utilizan los países que hablan la misma lengua?
- ¿Cual es el sentimiento y la polaridad expresada en los títulos?
- ¿Podemos modelar en número de veces que un vídeo se ve y establecer un modelo predictivo?

Estos análisis son de gran relevancia actualmente ya que como todos sabemos, la industria de creación de contenido online genera una gran cantidad

de dinero y público. Saber navegar éste mercado puede ser muy interesante para cualquier marca o persona que quiera generar beneficios o atraer público a través de YouTube.

2. Integración y selección de los datos de interés a analizar

Tal y como se menciona en la sección anterior este conjunto de datos extraídos de *Kaggle* cuenta con 10 datasets correspondientes a datos para la plataforma *YouTube* para diez países diferentes. Los datasets de los diferentes países cuentan con los mismos atributos. En general, los datasets también presentan un fallo en la codificación de los caracteres que hace que las columnas con texto cuyos caracteres no están dentro del alfabeto inglés, no se cargen correctamente. Debido a esto, se han seleccionado 4 datasets atendiendo a una serie de criterios:

- Podemos leer entender el idioma del país donde provienen los datos. De este modo, podemos corregir problemas de codificación de caracteres en un tiempo razonable.
- Representan zonas geográficas de varios continentes, en caso de que queramos hacer comparaciones regionales.
- Al menos dos datasets deben provenir de una regiones diferentes con la misma lengua. Esto permite hacer comparaciones en el uso del lenguaje.

Atendiendo a estos criterios hemos seleccionado 4 datasets sobre los que aplicaremos los mismo algoritmos. Todos los datos serán tratados de la misma forma. En caso de que algún dataset requiera un tratamiento especial en algún punto se especificará el motivo.

Los 4 datasets contienen datos para Alemania, Reino Unido, Mexico y EEUU.

2.1. Descarga de datos

Puesto que los datos ya contienen error de codificación de los atributos de texto y queremos evitar cualquier posible corrupción de los datos al pasar por una máquina intermedia (nuestros ordenadores), vamos a realizar una conexión con Kaggle a través de su API utilizando nuestro fichero `kaggle.json`

```
[ ] uploaded = files.upload()

Choose Files No file chosen Upload widget is only available when the cell has been executed in

[ ] # Since the data is a line of characters, it is possible to use:
user_key=uploaded['kaggle.json'].decode("utf-8", errors="ignore")

user = user_key[13:28] #Select the righth position for your name
key= user_key[37:-2] #Select the righth positions for the key

#Other method
user_key_2 = pd.read_json('kaggle.json', lines=True)

[ ] url="https://www.kaggle.com/datasnaek/youtube-new?select=USvideos.csv"

[ ] os.environ['KAGGLE_USERNAME'] = user # username from the json file
os.environ['KAGGLE_KEY'] = key # key from the json file
!kaggle datasets download -d datasnaek/youtube-new # api copied from kaggle
```

Figura 1: Carga de datos desde *Kaggle*

dónde guardaremos nuestras claves personales y descargamos los datos directamente a nuestro *drive* de google desde dónde cargaremos los datos a nuestro notebook de *Google Colab*.

Una vez nos hemos descargado los datasets y los tenemos en nuestro cloud, vamos a descomprimir el fichero y cargar los distintos datasets en un array de datasets que contendrá los diferentes datasets de cada mercado.

3. Limpieza de datos

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Para nuestros casos no hay valores nulos, como se puede ver en la siguiente figura 3 izquierda:

Los posibles elementos vacíos en los atributos de texto se tratan de forma separada durante la limpieza de texto.

Cuando buscamos para valores *NA* (figura 3 derecha), vemos que los valores vacíos se encuentran en la columna *description* (atributo texto), de la que no extraemos información en nuestro análisis. Aún así, no ha habido motivo de tratar estos valores directamente ya que durante otros procesos de limpieza, las obsevaciones que contienen *NAs* han sido eliminadas por otros motivos. También cabe mencionar que antes de empezar nuestro análisis

```
[ ] # Unzip the data

zf = zipfile.ZipFile(
    '/content/drive/MyDrive/Data_sets/YOUTUBE/youtube-new.zip',
    'r'
)

[ ] # Germany

videos_DE = zf.open(zipfile.ZipFile.namelist(zf)[3])
data_DE = pd.read_csv(videos_DE, encoding='maccentraleurope')

# Great Britain

videos_GB = zf.open(zipfile.ZipFile.namelist(zf)[7])
data_GB = pd.read_csv(videos_GB, encoding='iso-8859-1')

# Mexico

videos_MX = zf.open(zipfile.ZipFile.namelist(zf)[15])
data_MX = pd.read_csv(videos_MX, encoding='latin1')

# United States of America

videos_US = zf.open(zipfile.ZipFile.namelist(zf)[19])
data_US = pd.read_csv(videos_US, encoding='iso-8859-1')

[ ] data=[data_DE, data_GB, data_MX, data_US]
```

Figura 2: Descomprimiendo datos desde *Google Drive*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 16 columns):
#   Column                Non-Null Count  Dtype
---  -
0   video_id               40949 non-null object
1   trending_date          40949 non-null object
2   title                  40949 non-null object
3   channel_title          40949 non-null object
4   category_id            40949 non-null int64
5   publish_time           40949 non-null object
6   tags                   40949 non-null object
7   views                  40949 non-null int64
8   likes                  40949 non-null int64
9   dislikes               40949 non-null int64
10  comment_count          40949 non-null int64
11  thumbnail_link         40949 non-null object
12  comments_disabled      40949 non-null bool
13  ratings_disabled       40949 non-null bool
14  video_error_or_removed 40949 non-null bool
15  description            40379 non-null object
dtypes: bool(3), int64(5), object(8)
memory usage: 4.2+ MB
```

```
for i in range(4):
    print(data[i].isna().sum())

video_id                0
trending_date           0
title                   0
channel_title           0
category_id             0
publish_time            0
tags                    0
views                   0
likes                   0
dislikes                0
comment_count           0
thumbnail_link          0
comments_disabled       0
ratings_disabled        0
video_error_or_removed  0
description             1552
```

Figura 3: A la izquierda valores nulos. A la derecha valores *NA*

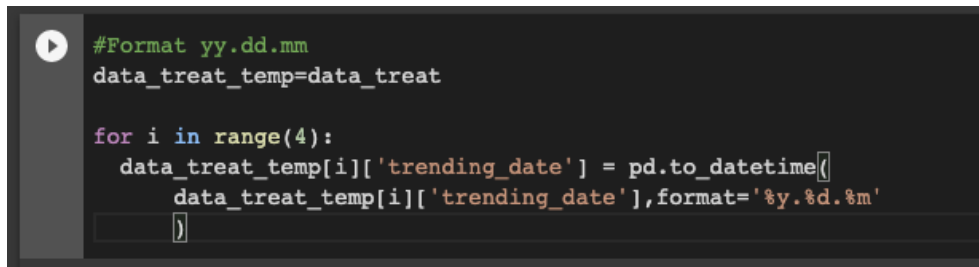
en *R* hemos eliminado explícitamente todas las observaciones que contenían valores *NA*.

3.2. Errores en en los códigos de tiempo

Una de las primeras errores que hemos podido detectar es que los datos referentes a códigos de tiempo se encontraban en formatos diferentes. Por lo que se ha unificado los formatos de tiempo y convertirlos datos de *object* a *datetime*.

3.2.1. Corrección trending date

Transformación de todas las columnas que contengan un valor de fecha a tipo datetime, empezando por la columna trending date.



```
#Format yy.dd.mm
data_treat_temp=data_treat

for i in range(4):
    data_treat_temp[i]['trending_date'] = pd.to_datetime(
        data_treat_temp[i]['trending_date'],format='%y.%d.%m'
    )
```

Figura 4: Conversión para *trending date*

3.2.2. Corrección publish time

Vamos a realizar la misma operación con la columna publish time.



```
# Format yy-mm-dd

for i in range(len(data_treat_temp)):

    for j in range(len(data_treat_temp[i]['publish_time'])):

        data_treat_temp[i].loc[j,'publish_time'] =pd.to_datetime(
            data_treat_temp[i].loc[j,'publish_time'][0:10],format='%Y-%m-%d'
        )
```

Figura 5: Conversión para *publish time*

3.3. Corrección textos

La parte de *language cleaning* es muy importante para nuestro proyecto, ya que tener textos limpios y analizables nos será útil para futuros análisis.

3.3.1. Limpieza general

En nuestro caso hemos definido diferentes funciones para corregir caracteres mal codificados, emoticonos, signos de puntuación que no necesitamos para el análisis del lenguaje natural, así como una función específica para tratar caracteres especiales del alfabeto alemán l. Hemos aplicado a las columnas: title, channel_title, tags y description. Las funciones se presentan en la figura 6.

```
#Latin characters
def replace_latin(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string
    rep = {'Ã': 'x', 'Ã': 'i', 'Ã': 'A', 'Ã': 'E', 'Ã': 'O', 'Ã': 'U', 'Ã': 'N', 'Ã': 'M',
          'Ã': 'O', 'Ã': 'O', 'Ã': 'U', 'Ã': 'U', 'Ã': 'I', 'Ã': 'I', 'Ã': 'S',
          'Ã': 'S', 'Ã': 'S', 'Ã': 'O', 'Ã': 'O', 'Ã': 'O', 'Ã': 'O', 'Ã': 'O', 'Ã': 'I'}

    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text) #correct latin

    return(text)

def remove_emotes(text):
    text_c = " ".join(filter(lambda x: x[0] != 'ð', text.split()))
    return(text_c)

[ ] def special_delete(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string
    rep = {'\n': ' ', '\x9c': '', '\xad': '', '\a': '', '\u': ''}
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)

def replace_german(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string
    rep = {'\s': 'ä', '\t': 'ü', '\u': 'ß', '\ö': 'ö', '\ø': 'é'}
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)

def replace_miss_spelling(text):
    # https://stackoverflow.com/questions/6116978/how-to-replace-multiple-substrings-of-a-string
    rep = {"exatlon": "Exatlón", "Exatlon": "Exatlón", "capitulo": "capítulo", "Capitulo": "capítulo"}

    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))

    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)

    return(text)
```

Figura 6: Corrección de texto

Además de las correcciones presentadas han sido necesarias, varias rondas de corrección de texto donde se han eliminado errores que no se habían capturado inicialmente. Todo esto se puede ver dentro del código en la sección *Some NLP-More Cleaning*

3.3.2. *Stop Words* para el *document-Term Matrix*

Con el fin de realizar las *World Clouds*, hay que eliminar del *document-Term Matrix* todas las *Stop Words*. El módulo *Count Vectorizer* de *sklearn* ya proporciona una lista de *Stop Words* para inglés pero debe ampliarse para otros lenguajes. De proyectos anteriores, ya disponíamos de listas de *Stop words* para otros idiomas, que hemos añadido, como se puede ver en la figura 7.

```
[ ] #German

File = open("/content/drive/MyDrive/Data_sets/YOUTUBE/german_stopwords_full.txt", "r")

list_of_german_stopWords = []
for line in File:
    stripped_line = line.strip()
    list_of_german_stopWords.append(stripped_line)

File.close()

[ ] #Spanish

File = open("/content/drive/MyDrive/Data_sets/YOUTUBE/stopwords-es.txt", "r")

list_of_spanish_stopWords = []
for line in File:
    stripped_line = line.strip()
    list_of_spanish_stopWords.append(stripped_line)

File.close()

[ ] #Merging Stop words
stop_words = text.ENGLISH_STOP_WORDS.union(manual_stop_words_1)
stop_words = stop_words.union(list_of_german_stopWords)
stop_words = stop_words.union(list_of_spanish_stopWords)

[ ] cv = CountVectorizer(stop_words=stop_words)
```

Figura 7: Añadiendo *Stop Words*

Además de estas *Stop Words* ha sido necesario añadir manualmente más palabras debido a los errores de codificación de los textos

3.4. Identificación y tratamiento de valores extremos

Para nuestro caso los valores extremos en visitas o cuentas de comentarios marcan los vídeos más populares, por lo que no parece acertado tratar estos datos como si de errores se tratasen, ya que estos valores extremos nos pueden indicar casos interesantes de estudio.

Por otra parte, al estudiar cuanto tiempo tarda un vídeo en convertirse en tendencia, hemos visto que hay vídeos que pueden tardar hasta once años en llegar a tendencia. Estos casos, si son outliers para nosotros.

Nuestro caso de estudio es, cómo las cosas que ocurrieron ayer afecta a los que será tendencia mañana y cuanto dura el impacto de estos hechos

en el tiempo. Después de analizar las deltas de tiempo hemos considerado razonable eliminar cualquier video que tarde más de 40 días en convertirse en tendencia por dos razones; Si algo tarda más de 40 días en convertirse en tendencia no proviene de vídeos de historias recientes. Los vídeos que tardan más de 40 días en convertirse en tendencia suponen menos del 1,5 % de la información total. En la figura 8 se presenta la método de eliminación de estos datos.

```
[ ] #check for the percent of information lost

for data in data_treat:

    x= (data['time_delta'][data['time_delta']<=40].count())/len(data['time_delta'])
    x=round(100-(x*100),2)
    print(x)

0.05
1.67
0.1
0.8

[ ] #Filtering

for i,data in enumerate(data_treat):

    data_treat[i]=data.loc[data['time_delta']<=40,:]
```

Figura 8: Eliminamos las observaciones con una delta de tiempo superior a 40 días y se comprueba la cantidad de información perdida

4. Análisis de los datos

4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

4.1.1. Vídeos con defectos

De entre todas las columnas destacan unas con valores lógicos: *comments_disabled*, *ratings_disabled* y *video_error_or_removed*, que están marcados indicando algún tipo de carencia.

Con el fin de homogeneizar los datos, es decir, hacer que todos nuestros datos provengan de vídeos con las mismas características generales, vamos a eliminar las filas con valores *True* en cualquiera de estos atributos. Esto se puede ver en la figura 9. También se puede observar que en general se conserva más del 96 % de la información.

```
[ ] data_treat=[]

for dt in data:
    data_treat.append(dt[(dt['comments_disabled']== False) &
                        (dt['ratings_disabled']== False) &
                        (dt['video_error_or_removed']== False)])

[ ] for i in range(4):

    print(round((len(data_treat[i])/len(data[i]))*100,2))

96.33
97.94
97.62
98.24
```

Figura 9: Eliminamos los vídeos con defectos y comprobamos cuanta información se conserva

Debido a que estas columnas solo contienen valores *FALSE* después de eliminar los *TRUE*, ya no es relevante conservar estas columnas por lo que se eliminan (esto se puede ver en código en la sección *data cleaning*).

4.1.2. Eliminamos categoría sin información (*category = 29*)

No existe información sobre que categoría representa el id 29, por lo tanto eliminamos ésta del dataset. Otra razón para eliminarla es que al no saber que representa dicha categoría, aun en el caso que pudiésemos extraer alguna información para la categoría 29, no podríamos darle un sentido concreto

En la figura 10, se puede ver que después de eliminar los observaciones de esta categoría, preservamos más del 95 por ciento de la información.

4.1.3. Generación de métricas y análisis de datos

Nuestros datos en bruto pueden carecer de sentido sin una referencia; ¿Son dos vídeos con el mismo número de visitas igual de relevantes? o Para dos videos con el mismo número de likes, ¿Podemos pensar que han despertado el mismo interés?. Otro ejemplo sería tenemos videos con un gran número de visitas pero pocos comentarios (lo que llamaríamos un click bait), o un video con pocas visitas pero un gran número de comentarios.

Para poder solventar inconveniente de que carecemos de referencias y poder analizar correctamente los vídeos, creamos una serie de métricas relativas

```
[ ] for i in range(4):

    data_treat[i]=data_treat[i][(data_treat[i]['category_id']!= 29)]
    data_treat[i].reset_index(drop=True, inplace=True)#Reset indexes

[ ] for i in range(4):

    print(round((len(data_treat[i])/len(data[i]))*100,2))

95.77
97.71
97.25
98.11
```

Figura 10: Eliminamos la categoría 29 y comprobamos el porcentaje de información conservada

que nos ayudarán a realizar análisis más de los vídeos.

Sentimiento relativo:

La métrica *relative video sentiment* (rvs) se calcula utilizando la fórmula $(likes-dislikes)/sum(likes+dislikes)$ y nos ayudará a hacernos una idea de el sentimiento que genera un video sobre 1. Si el sentimiento que genera el vídeo es positivo, el valor de rvs será positivos y si tiene más *dislikes* el valor de rvs será negativo. Se puede ver explícitamente el proceso seguido en la figura 11

```
[ ] for dt in data_treat:
    dt['rvs']=round((dt['likes']-dt['dislikes'])/(dt['likes']+dt['dislikes']),3)
```

Figura 11: Ecuación para generar rvs

Haciendo un plot de las frecuencias obtenidas sobre esta métrica, nos damos cuenta de que los videos que llegan a ser *trending*, tienden a tener más *likes* que *likes*. De hecho la mayoría de vídeos que llegan a *trending* tienen al menos un 75 % de *likes* frente a *dislikes*. Esto se puede ver en la figura 12.

Métrica relevancia:

Sabemos que para los usuarios comentar un video conlleva más esfuerzo que hacer click a *like* o *dislike*. También entendemos que un video tiene mayor cantidad de comentarios cuando los viewers tienen algo que decir sobre el tema o algún comentario a provocado controversia.

Para medir esto, hemos creado un atributo *rel_relevance*, el cual se calcula con la fórmula $comment_count/(likes+dislikes)$. Cuando su valor tiende a 1 asumimos que el video es relevante para los *viewers* y cuando es mayor que 1

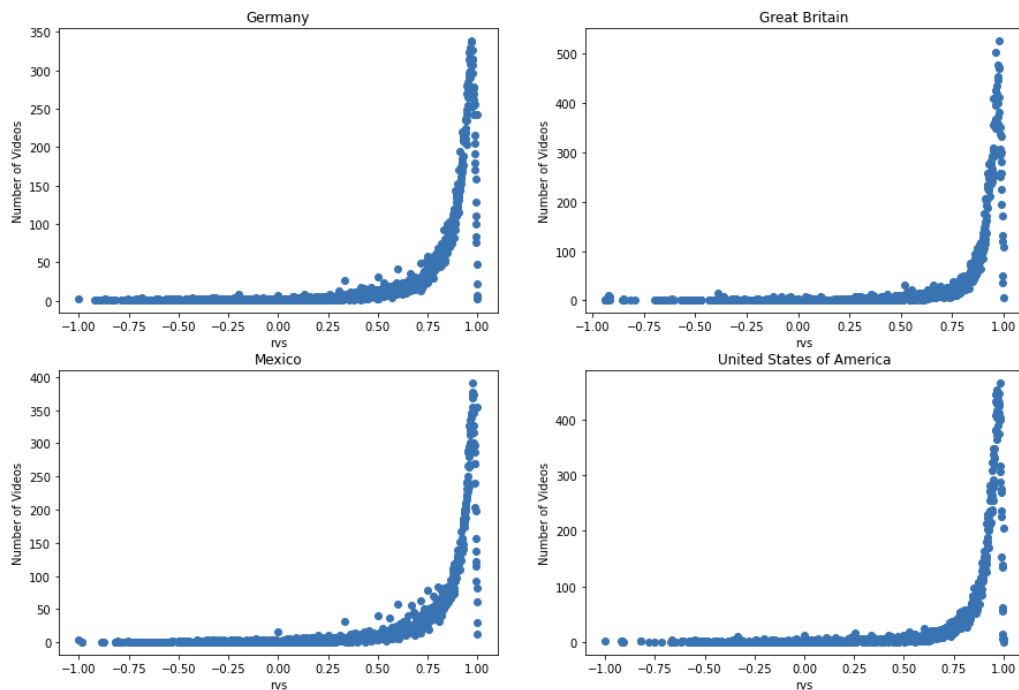


Figura 12: Número de ocurrencias por rvs

quiere decir que es muy relevante. La generación de la métrica se puede ver la figura 13.

```
[ ] for dt in data_treat:
    dt['rel_relevance']=round((dt['comment_count'])/(dt['likes']+dt['dislikes']),3)
```

Figura 13: Ecuación para generar la métrica de relevancia relativa

Haciendo un plot de las frecuencias de este valor (figura 14), podemos observar que la mayor parte de los videos tiene un valor menor que 1. De hecho, observamos que los vídeos que alcanzan *trending* tienen a tener un ratio menor que 0.5, alcanzando un pico claro alrededor 0.25.

Esto nos da a entender que los vídeos que alcanzan *trending* no suelen ser muy polémicos en general, lo cual tiene sentido ya que suelen ser vídeos que atraen a un público general.

Métrica sentiment engagement:

Por último vamos a generar 3 atributos relacionados con el *sentiment engagement* de los vídeos, éstos serán el *positive engagement*, *negative engagement* y *overall engagement*.

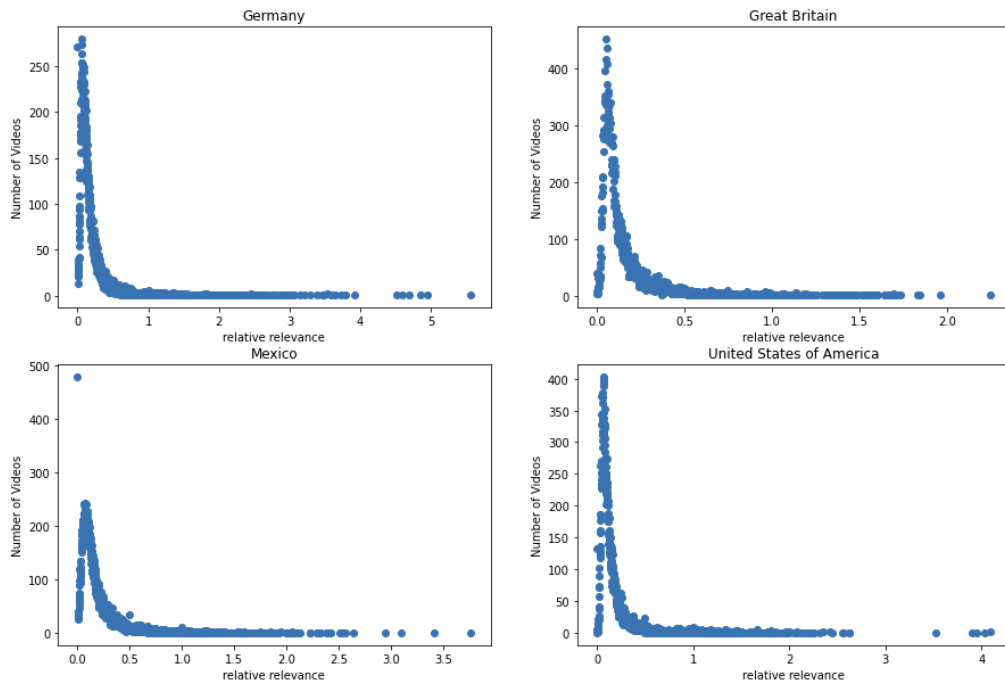


Figura 14: Número de ocurrencias para la relevancia relativa

Estas métricas se calculan como:¹

- *Positive engagement* se calculará utilizando la fórmula: $likes/views$
- *Negative engagement* se calculará utilizando la fórmula: $dislikes/views$
- *Overall engagement* se calculará utilizando la fórmula: $(likes-dislikes)/views$

Al representar sus frecuencias (figura 15) podemos ver que los vídeos *trending* tienen un número de *likes* o *dislikes* muy pequeño en comparación con las visualizaciones. De hecho, la mayoría de ellos apenas tienen un 10% de *likes* en comparación con las views. Y la mayoría de las views son *likes*.

Esto último es más evidente en la curva para *Overall engagement* que para todos los casos, está más desplazada hacia el lado de los valores positivos.

¹Los cálculos se han realizado de la misma forma que se ha mostrado en la demás métricas. Los detalles pueden verse en el notebook, en el apartado de métricas.

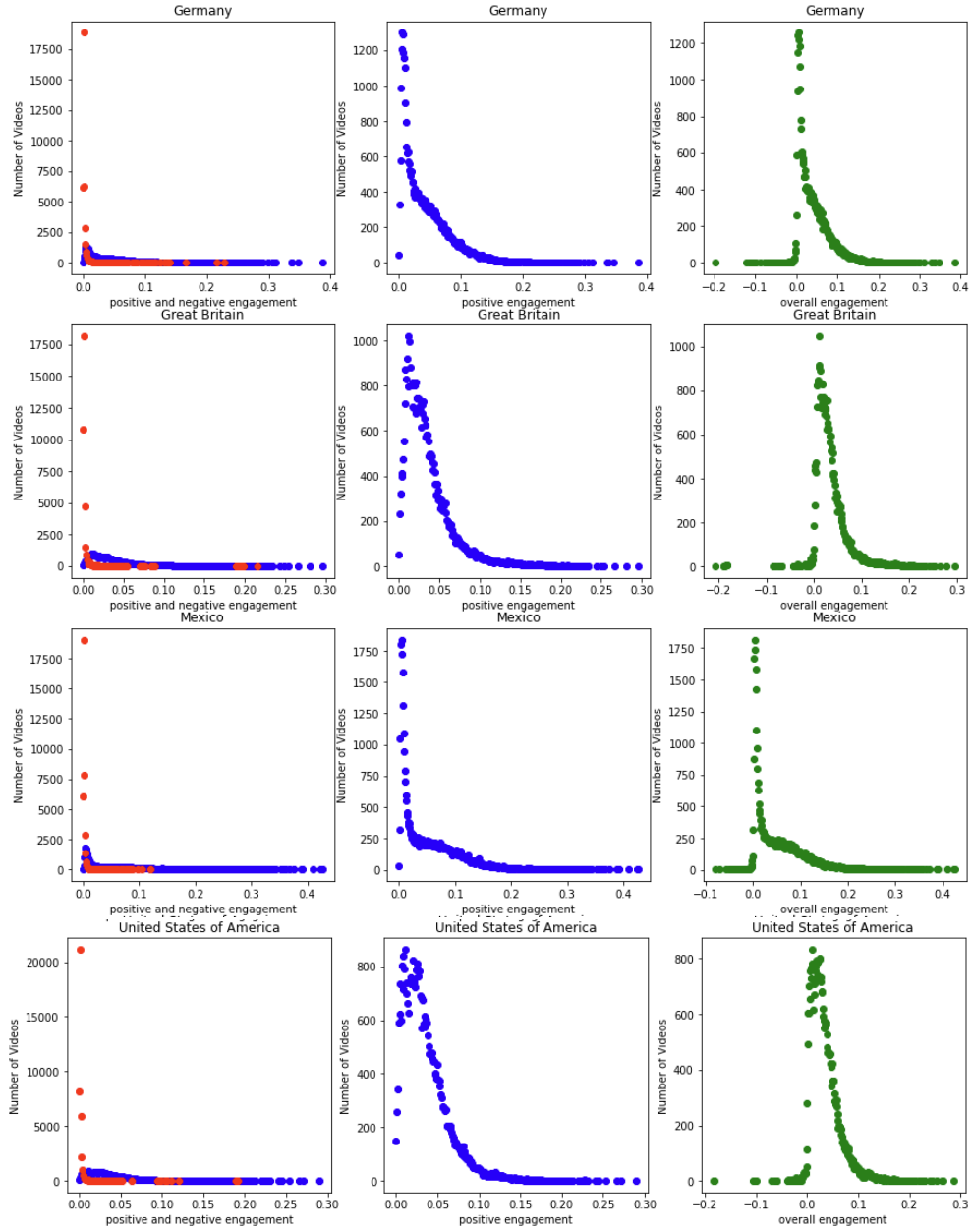


Figura 15: Número de ocurrencias para cada *sentiment engagement*

4.2. Comprobación de la normalidad y homogeneidad de la varianza.

Nuestro daset presenta dos dificultades a la hora de comprobar si los datos y las métricas siguen una distribución normal. Por una parte el gran número

	views	likes	dislikes	comment_count	rvs	rel_relevance	overall_sentiment_engagement
Germany	0	0	0	0	0	0	0
Great Britain	0	0	0	0	0	0	0
Mexico	0	0	0	0	0	0	0
United States of America	0	0	0	0	0	0	0

fviews	flikes	fdislikes	fcomment_count	frvs	frel_rel	foverall_sen_eng
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	1	0
1	1	1	1	0	0	0

Figura 16: Comprobación de normalidad

de datos hace inviable la aplicación del test de normalidad de *Shapiro Wilk*, por otra parte, al existir muchos valores repetidos para un atributo el test de *Anderson Darling* no es fiable ya que siempre devolverá como resultado el rechazo de la hipótesis nula. Por este motivo nos centramos en los valores de *Kurtosis* y *Skewness*. Para estos test, típicamente se considera que una distribución es normal cuando el valor de *Kurtosis* está entre -2 y $+2$ y para *skewness* los valores deben caer dentro del intervalo entre $-0,8$ y $+0,8$.

Hemos creado una función *normal check* que nos devuelve 1 cuando los datos de un atributo siguen una distribución normal (en las condiciones especificadas arriba) y 0 cuando no la sigue. La hemos aplicado sobre todas las columnas de todos los datasets y los datos los hemos visualizado como un dataframe que se copia abajo (los detalles de la función se encuentran en el notebook).

Podemos comprobar que tanto los valores de los datos en crudo, como las métricas que hemos implementado no siguen una distribución normal (figura 16). Con el fin de solucionar este problema y conseguir una distribución homogénea de la varianza, hemos optado por utilizar en primera instancia una transformación de *Yeo Jhonson*, puesto que nos permite aplicar el mismo código sobre todos los atributos (una transformación de *Yeo Jhonson* a diferencia de *Boxcox* nos permite trabajar con valores negativos y ceros).

Como se puede ver los datos en crudo se han podido transformar de forma

que ahora siguen una distribución normal (figura 16). Los datos transformados se guardan en las columnas *fvviews*, *likes*, *dislikes* y *fcommentl_count*. Además los valores de lambda también se guardan para poder deshacer la transformación (se pueden ver todos los detalles en el notebook en el apartado *Power transformation and descriptive statistics for the metrics*).

Sin embargo, las métricas no siguen todavía una distribución normal. Intentamos corregir este hecho hemos realizado una transformación por cuantiles sobre las métricas. Los nombres asociados a los datos de las métricas transformados por cuantiles son : *q_rvs*, *q_rel_rel* y *q_overall_sen_eng*. Una vez hechas las transformaciones, volvemos aplicar la función implementada para el cálculo de la normalidad de los atributos y representamos los resultados en un dataframe que se puede ver en la figura 17.

	q_rvs	q_rel_rel	q_overall_sen_eng
Germany	0	0	1
Great Britain	0	0	1
Mexico	0	0	0
United States of America	0	0	1

Figura 17: Transformación por cuantiles para las métricas

Se ve una discrepancia en los datos de *México*. Hacemos un gráfico de densidades para la los datos *q_overall_sen_eng*, es decir los datos que cuya variable ha sufrido una transformación por cuantiles.

En la curva de la figura 18, no podemos apreciar ninguna diferencia significativa entre las curvas para cada país.

En conclusión ahora contamos con los datos transformados vía *YeoJhonson* que siguen una distribución normal y la métrica *overall_sent_eng* que transformada por cuantiles también sigue una distribución normal por lo que podremos aplicar test de correlación, modelos de regresión y otros test estadísticos para métricos.

Como nota final también se ha investigado la normalidad de las deltas de tiempo y su distribución no es normal. Pero se ha podido alcanzar una distribución normal usando transformaciones exponenciales (el código con los cálculos se encuentran en el notebook; apartado *How long does it take for a Video to get trending?* sub apartado *power transformation*).

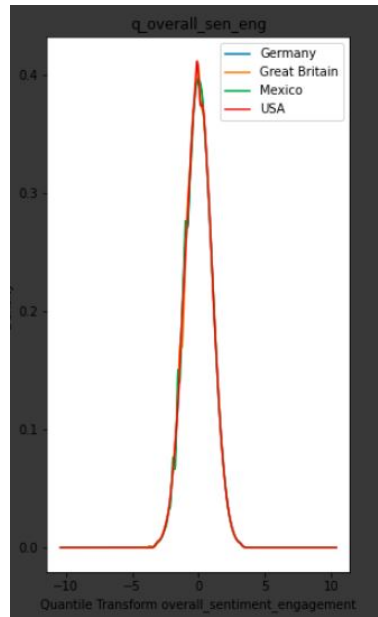


Figura 18: Gráfico de densidades para $q_overall_sen_eng$

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

4.3.1. Método 1: Correlación

Una vez normalizado los datos podemos aplicar el método de correlación de *Pearson*. En la figura 19 es interesante ver que los atributos transformados presentan una alta correlación. Si ahora nos fijamos en los atributos sin transformar esta correlación, aunque alta, no es tan marcada como en los atributos transformados. La alta correlación también apunta al hecho que es factible esperar encontrar una relación lineal entre atributos transformados.

4.3.2. Método 2: Modelos de regresión

En este punto nuestro objetivo es construir un modelo que nos permita predecir el número de *views* en función de otras variables. En este sentido, como muestra la figura 20, los atributos se han dividido en tres grupos:

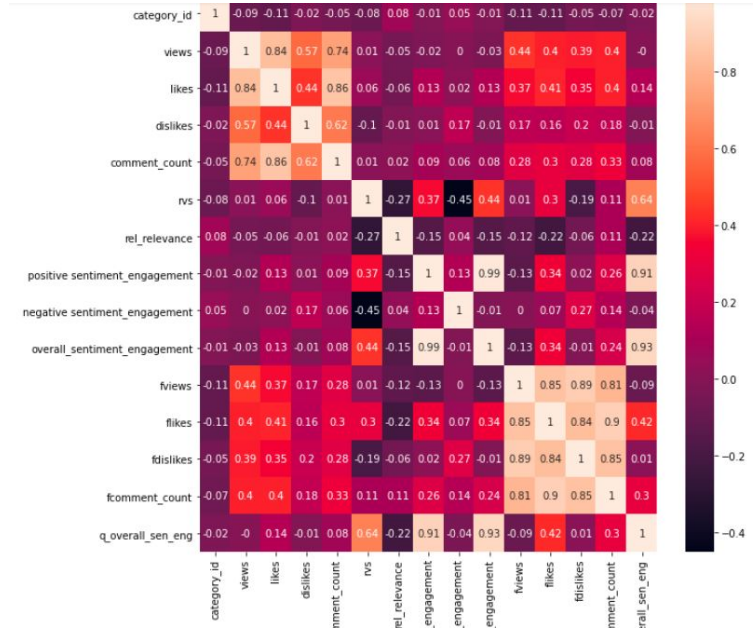


Figura 19: Gráfico de correlaciones para todos los atributos

```
[ ] %%R
not_trans_raw<-c('views','likes','dislikes','comment_count')
not_trans_met<-c('views','rvs','rel_relevance ','positive_sentiment_engagement',
'negative_sentiment_engagement','overall_sentiment_engagement')
trans_atr<-c('fviews','flikes','fdislikes','fcomment_count','q_overall_sen_eng')
```

Figura 20: División de atributos para el cálculo de modelos de regresión

- *not_trans_raw*: Donde se hace uso de los datos en crudo para intentar predecir el número de *views*
- *not_trans_met*: Donde se hace uso de las métricas para intentar predecir el número de *views*
- *trans_atr*: Donde se hace uso de los datos transformados para poder predecir el número de *views* (transformadas).

A continuación se deja el 85 % de los datos para entrenar los modelos, y el resto para hacer el test. Los modelos que se han implementado se pueden ver en la figura 21.

Los valores que obtenemos para R^2 -ajustado son: 0,752 para *model_raw_1*, 0.001 para *model_metric_1* y 0.953 para *model_trans_1*.

```

%%R
set.seed(123)

train.indice <- sample(1:dim(r_raw)[1],(dim(r_raw)[1]*85)/100)

train_raw <- r_raw[ train.indice,]
train_metric<-r_metric[train.indice,]
train_trans<-r_trans[train.indice,]

test_raw <- r_raw[-train.indice,]
test_metric<-r_metric[-train.indice,]
test_trans<-r_trans[-train.indice,]

[ ] %%R

model_raw_1 <- lm(views ~ ., data = train_raw)
model_metric_1 <- lm(views ~ ., data = train_metric)
model_trans_1 <- lm(fviews ~ ., data = train_trans)

```

Figura 21: División de los datos en train y test e implementación de modelos

A tenor de estos resultados se escoge como mejor modelo: *model_trans_1*. Este modelo usa los atributos que han sufrido una transformación para que los atributos tengan una distribución normal. Los atributos también presentaban una alta correlación. Con el fin de otorgar sentido al *intercept*, también se hace el modelo con los datos centrados y escalados. De esta forma el *intercept* representa el valor de *fviews* cuando todos los atributos tienen un valor igual a su media. En cualquier caso, el valor de R^2 ajustado se mantiene constante tanto para el modelo que usa los datos escalados como el que no. Por este motivo mostramos el *summary* del modelo con los datos escalados y centrados (figura 22) .

En la figura 22, se puede ver que el atributo *fdislikes* no supera el valor típico para otorgarle relevancia estadística. Por lo que hemos repetido el modelo, pero esta vez sin el atributo *fdislikes*. Al modelo que cuenta con la variable *fdislikes* es al que llamamos modelo completo y al modelo que no cuenta con la variable *fdislikes* lo llamamos reducido. Tanto el modelo completo como el reducido tienen el mismo valor de R^2 ajustado. Con el fin de comprobar formalmente si un modelo es mejor que otro se ha hecho una prueba *anova* sobre ambos modelos (figura 23).

En la figura 23 se puede ver que el valor de p es lo suficientemente alto como para no poder rechazar la hipótesis nula. Por lo no podemos decir que los dos modelos sean diferentes. En este caso elegimos el modelo reducido ya que tienen menos variables y nos ofrecerá los mismo resultados.

Por último hemos querido comprobar la capacidad predictiva de nuestros modelos, y se ha calculado el valor RMSE (desviación de mínimos cuadrados),

```

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min       1Q   Median       3Q      Max
-2.9024 -0.1073 -0.0029  0.1216  2.4695

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.8118693   0.0009869  7915.467 < 2e-16 ***
flikes        1.0083187   0.0034795   289.786 < 2e-16 ***
fdislikes     -0.0047087   0.0028233    -1.668  0.0954 .
fcomment_count -0.0184368   0.0024237    -7.607 2.88e-14 ***
q_overall_sen_eng -0.5901333   0.0019171  -307.820 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1804 on 33400 degrees of freedom
Multiple R-squared:  0.9532,    Adjusted R-squared:  0.9532
F-statistic: 1.701e+05 on 4 and 33400 DF,  p-value: < 2.2e-16

```

Figura 22: Resumen del modelo *model_trans_1*

```

Model 1: fviews ~ flikes + fdislikes + fcomment_count + q_overall_sen_eng
Model 2: fviews ~ flikes + fcomment_count + q_overall_sen_eng
      Res.Df  RSS Df Sum of Sq    F Pr(>F)
1    33400 1086.7
2    33401 1086.8 -1 -0.090499  2.7815 0.09537 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figura 23: Anova sobre el modelo completo y el reducido

tanto para el modelo de datos en crudo como para el modelo reducido (que usa los atributos transformados), sobre sus respectivos *test sets*. Además con el fin de poder establecer una comparación entre ambos valores, el valor de RMSE se ha dividido por los valores máximos de *views* y *fviews* de los *test sets*.

Para el caso del modelo con datos en crudo hemos encontrado un valor de RMSE de 925483 y al dividirlo por su máximo 19972 (la desviación es casi 20000 mayor que la propia medida).

Para el modelo reducido tenemos un RMSE de 0,1830192 y al dividirlo por el valor máximo 0,0553140. El hecho de que el valor relativo sea menor que uno ya es indicativo que la desviación es inferior a la medida.

4.3.3. Método 3: Pruebas de contraste de hipótesis

En este punto nos preguntamos que conjunto de categorías de vídeos tienen medias de número de visualizaciones similares estadísticamente (no se

puede rechazar la hipótesis nula). Una forma de comprobar esto es realizar comprobaciones por pares (hecho en el código del notebook), sin embargo, tal y como recoge el material que proporciona la UOC para la asignatura de estadística avanzada corremos el riesgo de aceptar algún resultado significativo incluso cuando todos no lo son. Una las alternativas que se proponen es usar una corrección conservadora (Bonferroni). El paquete *agricolae* que implementa esta corrección tiene además la ventaja que ya nos codifica con las mismas letras los grupos para los que no existe diferencia. En la figura 24 se puede ver el resultado para Mexico (aunque en el código del notebook están los resultados para todos los países).

	data	groups
10	8.214677	a
20	8.134795	ab
23	8.120238	b
17	8.008740	c
15	7.883022	cd
1	7.878937	d
19	7.854130	d
24	7.847919	d
2	7.807444	d
28	7.797578	d
25	7.641420	d
26	7.613934	d
22	7.522938	e
27	7.366742	f
43	6.854178	f

Figura 24: Agrupación de categorías similares codificadas por letras para Mexico

5. Representación de los resultados a partir de tablas y gráficas.

5.1. Cuánto tarda un video en llegar a Trending

Como ya hemos apuntado en la sección sobre el tratamiento de *outliers*, más del 95 % de los vídeos llegan a *trending* en los primeros 40 días. Con el fin de poder visualizar más claramente los deltas de tiempo, en la figura 25

mostramos como hemos dividido los vídeos en intervalos de tiempo según si tardan entre $[0, 1]$, $(1,2]$, $(2,5]$, $(5,7]$, $(7, 10]$, $(10, 15]$, $(15,30]$ y $(30, 40]$ días en llegar a *trending*. Luego hemos calculado el porcentaje de los vídeos con respecto al total.

```
#checking for the percent of videos that become trending in at least 30 days

times=[0,1,2,5,7,10,15,30]

times_list=[]

for data in data_treat:

    time_list_temp=[]

    for i, time in enumerate(times):
        try:
            x=(data['time_delta'][(data['time_delta']>=time) & (
                data['time_delta'] < (times[i+1])) ].count()/(len(data['time_delta']))
            )

            x=round(x*100,2)

            time_list_temp.append(x)
        except:

            pass

    times_list.append(time_list_temp)

[ ] title_label=['Germany','Great Britain','Mexico','United States of America']
times=[0,1,2,5,7,10,15,30]

plt.figure(figsize=(15,10))

index_=1

for tm in times_list:

    plt.subplot(2,2,index_)
    plt.scatter(times[1:],tm)
    plt.title(title_label[index_-1], loc='center')
    plt.xlabel("Time delta")
    plt.ylabel("Percent of Videos that get trending")
    index_+=1

plt.show()
```

Figura 25: Implementación de la división del tiempo en intervalos y cálculo de porcentaje

Los datos mostrados representan en la figura 26 un 98% de los videos. Estos resultados son muy interesantes ya que nos dan una idea de cuando

tarda algo en hacerse popular en un los diferentes paises, cada cuanto utilizan la plataforma los usuarios en los diferentes paises, o cuanto tarda un usuario en ver algo fuera de sus intereses habituales. Por ejemplo en Gran Bretaña es razonable esperar que un vídeo pueda tardar hasta 30 días en hacerse *trending*. Para Alemania y Mexico, si el vídeo no se ha hecho *trending* en cinco días, sus posibiliades de llegar a *a trending* caen rápidamente.

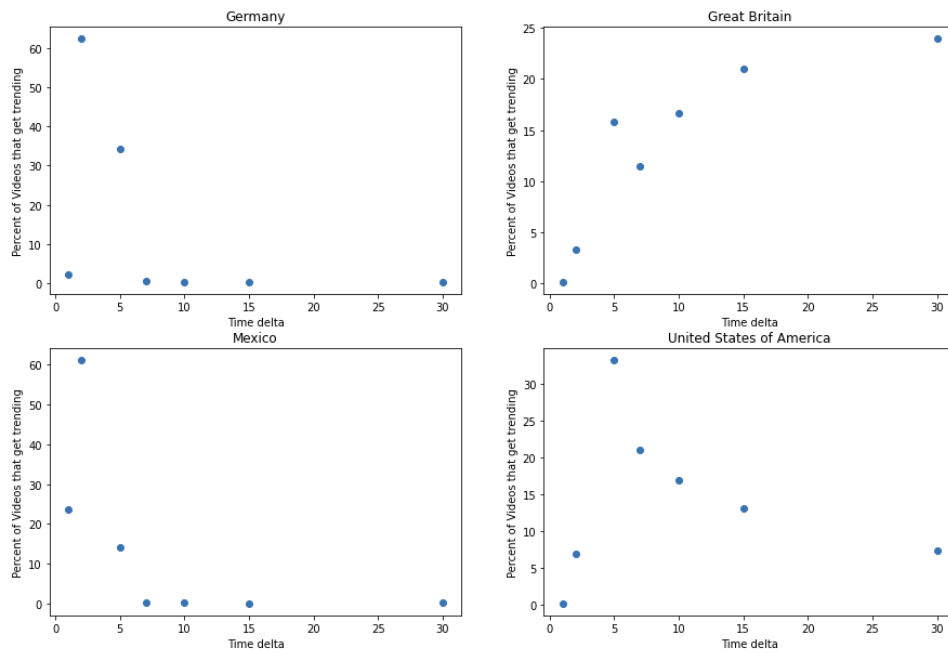


Figura 26: Intervalos de tiempo y porcentaje de vídeos

5.2. Qué categorías tienen más éxito en cada país

Para realizar visualizaciones correctas sobre las categorias de los videos, vamos crear una columna con el nombre de la categoría del video (ahora mismo solo teniamos sus ids).

```

import json

# Read JSON ID
categories_DE = zf.open(zipfile.ZipFile.namelist(zf)[2])
json_dict = json.load(categories_DE)

# Create dict id => name
categories_dict = {}
items = json_dict["items"]
for item in items:
    categories_dict[int(item["id"])] = item["snippet"]["title"]

cat_keys = categories_dict.values()

# Map and create categories name column
for df in data_treat:
    df["category_name"] = df["category_id"].map(categories_dict)

```

Una vez tenemos la columna creada, vamos a obtener el numero total de videos en trending por categoria en cada mercado.

```

[ ] # We get the list of categories
title_label=['Germany','Great Britain','Mexico','United States of America']
counter = 0
index_brais = 1
num_dict = {}
fig = plt.figure(figsize=(20,30))

# Iterate through each dataset
for df in data_treat:
    categories_list = df.category_name.unique()
    categories_df_list = []
    rvs_dict = {}
    num_dict = {}

    # For each category, we get the number of videos
    for category in categories_list:
        category_df = df.loc[df['category_name'] == category]
        rvs_dict[str(category)] = category_df.rvs.mean()
        num_dict[str(category)] = category_df.shape[0]

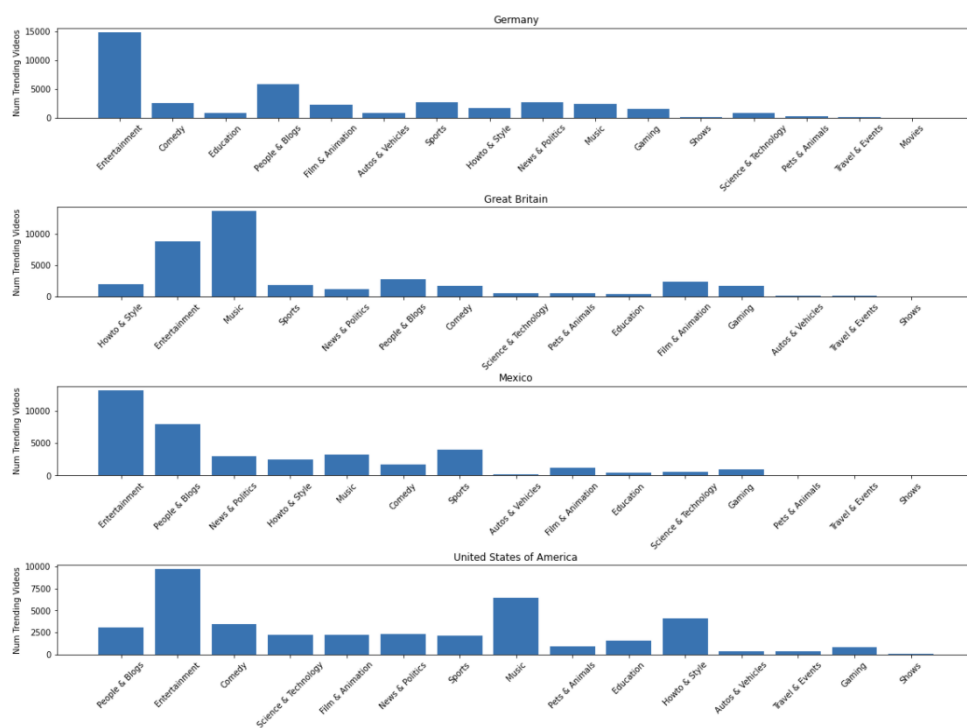
    # Show the count of videos per category
    plt.subplot(8,1,index_brais)
    plt.subplots_adjust(hspace=1)
    keys = num_dict.keys()
    values = num_dict.values()
    plt.title(title_label[counter])
    plt.ylabel("Num Trending Videos")
    plt.xticks(rotation=45)
    plt.bar(keys, values)
    index_brais +=1
    counter +=1

fig.show()

```

Con este plot vamos a ser capaces de ver la distribución de las vistas por cada categoría. Como datos más relevantes que se pueden observar, los videos de categoría entretenimiento son los que suelen alcanzar trending con más asiduidad.

Otro dato que me ha parecido muy interesante es como los videos musicales son muy populares en EEUU y UK, mientras que en Mexico o Alemania no son casi relevantes. Ésta información podría ser de gran utilidad para artistas que intentan crecer en diferentes regiones o mercados.



5.3. NLP

Con el fin de no sobrecargar el trabajo hemos eliminado del ejercicio escrito el cálculo de las 50 palabras más relevantes para cada atributo. Así como el análisis de sentimiento para los títulos y el de polaridad. Sin embargo todo el proceso está detallado en el notebook en el apartado *some NLP*. Cabe destacar que hemos encontrado varios problemas en análisis de los textos del dataset Alemán. Esto se debe a que en Alemania hay una fuerte componente de inmigración turca, y muchos videos tienen títulos, comentarios y tags en turco, eso unido a la pobre codificación del texto de base y que no entendemos

turco nos dificulta extraer resultados satisfactorios del dataset alemán.

5.3.1. Worcloud de n-gramas

A través del análisis del *document-term Matrix* y utilizando el código que se presenta en la figura 27 hemos podido extraer *word clouds* con sentido para los atributos de texto de cada país.

```
[ ] wc = WordCloud(stopwords=stop_words, background_color="white", colormap="Dark2",
                    max_font_size=150, random_state=42)

[ ] plt.rcParams['figure.figsize'] = [16, 8]

countries = ['Germany', 'Graet Britain', 'Mexico', 'USA']

for index, country in enumerate(data_dtm_trans[2].columns):
    wc.generate(List_of_txt_frames[2].tags[country])

    plt.subplot(2, 2, index+1)
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.title(countries[index])

plt.suptitle('Tags')
plt.show()
```

Figura 27: Código para generar las *word clouds*

Las **tags** se pueden ver en la figura 28.

Destacar en el Mexicano, que podemos ver la TV Azteca y exatlon, que es un programa de gran éxito en el país (de TV Azteca). En EEUU parece que *makeup tutorial* es importante. En Gran Bretaña *Star Wars* o *Black panther* (en este momento cabe destacar que estos datos son del 2017)

Los **títulos** de los vídeos los podemos ver la figura 29.

A destacar que la palabra Trump aparece más en los títulos de Alemania que en EEUU. Los vídeos musicales parecen ser muy populares. Los vídeos de fútbol y otros deportes son mucho más populares en México que en el resto de países. Los vengadores y Star Wars tienen bastante impacto en UK y EEUU.

Por último, en la figura 30, mostramos las palabras más utilizadas en las **descripciones** de los vídeos.

Como se puede observar, lo que más se ven son nombres de redes sociales y webs, esto se debe a que la mayoría de creadores de contenido introducen

Tags



Figura 28: *Word clouds* para el atributo tag por país

Video Title



Figura 29: *Word clouds* para el atributo título por país

sus redes sociales en la descripción. Para poder obtener información relevante de ésta wordcloud necesitamos realizar más tareas de limpieza.

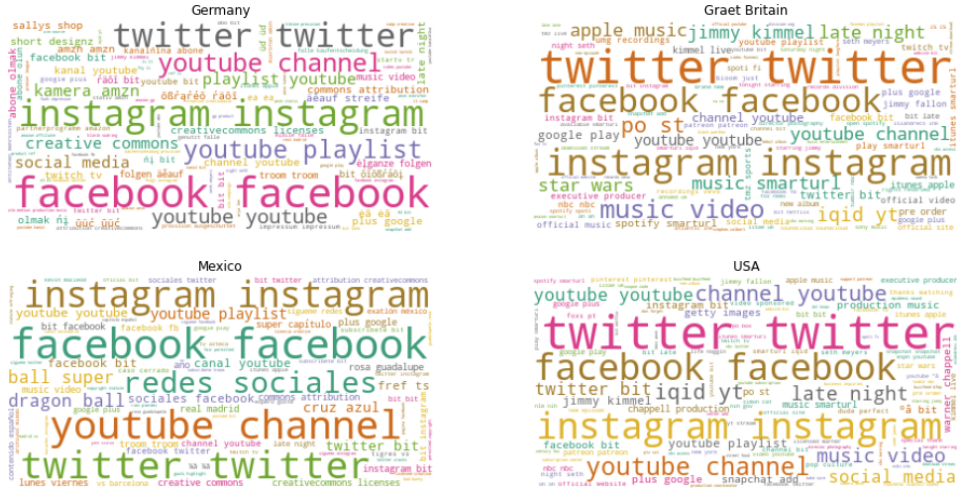


Figura 30: *Word clouds* para el atributodescripción por país

5.3.2. Uso de vocabulario

Una de las cuestiones que nos planteamos al presentar los objetivos de la práctica es si podíamos encontrar diferencias en el uso del vocabulario de dos países donde se hable la misma lengua.

Para realizar este análisis se han contado todas las palabras que al menos ocurren una vez en el *document term Matrix* de los datasets de Gran Bretaña y EEUU. Como podemos ver en la figura 31 de la izquierda en ambos países se usan la misma cantidad de palabras únicas. Sin embargo cuando nos preguntamos por la diversidad del vocabulario, es decir, en que país se usa una variedad más amplia de palabras vemos que la diferencia entre Gran Bretaña y EEUU es bastante notable. En EEUU hay muchas más palabras que aparecen más e 100 veces, por lo que el vocabulario de EEUU se encuentra más concentrado en un número de palabras más reducido que si lo comparamos con el vocabulario de Gran Bretaña (figura 31 derecha).

	country	unique_words
0	Great Britain	35491
1	USA	35491

	country	unique_words
0	Great Britain	7980
1	USA	9500

Figura 31: Diferencias en vocabulario

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Siguiendo el esquema de preguntas que nos planteábamos al principio de la práctica:

- ¿Qué categorías son mas populares en cada mercado?
 - Si tuviésemos que generalizar, podemos decir para la mayoría de países Entretenimiento y Música son las que mas destacan.
 - Hemos descubierto que ésto varía mucho dependiendo del país, vemos como por ejemplo los videos musicales son de gran importancia en el mercado Estadounidense, pero no son casi relevantes en el Alemán.
- ¿Cuánto tarda un video en hacerse Trending?
 - Depende fuertemente el país. De los países investigados hasta 30 días es razonable para Gran Bretaña. Para el resto, después de cinco días de su publicación la cantidad de vídeos que llegan a *a trending se reduce exponencialmente*
- ¿Cuales son las palabras más utilizadas?
 - Aunque no está incluido en el texto si en el código. Se han generado diccionarios con las 50 palabras más utilizadas para los distintos atributos de texto de cada país. Como curiosidad la palabra que más aparece en los títulos de Gran Bretaña es Tyron: profesional Americano de artes marciales mixtas y comentarista deportivo.
- ¿Cuales son las tags más relevantes?
 - Cabe destacar *makeup tutorial* para EEUU por lo singular de la tag.
 - De forma general para Gran Bretaña y EEUU *Star Wars*, videos musicales, *Super bowl* y temas relacionados con los Vengadores
 - para Mexico principalmente Tv azteca o Real Madrid
- ¿Cuales son las temas más importantes?

- Cine en especial las dos franquicias mencionadas.
 - Música en especial hip hop y vídeos musicales
 - Deportes
 - También se pueden ver otros temas que van desde política a dragon Ball .
- ¿Existen diferencias en el vocabulario que se utilizan los países que hablan la misma lengua?
- Si, en EEUU el vocabulario se encuentra concentrado en número más reducido de palabras si lo comparamos con el caso de Gran Bretaña.
- ¿Cual es el sentimiento y la polaridad expresada en los títulos?

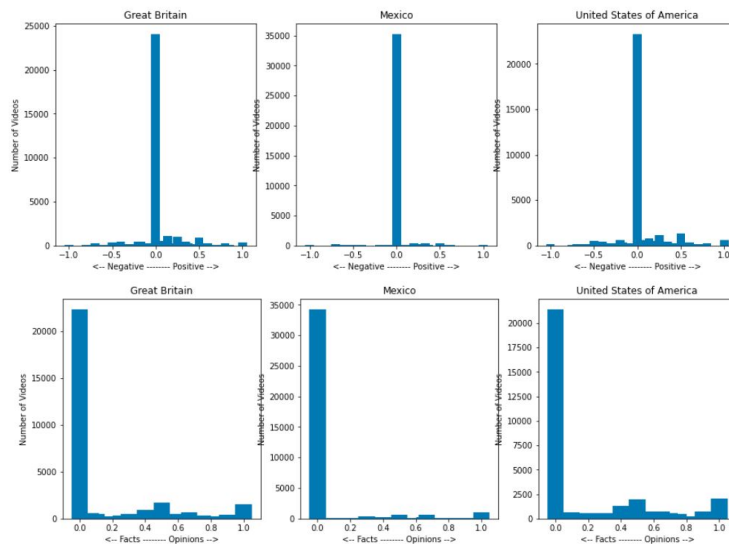


Figura 32: Sentimiento arriba, polaridad abajo

- Como se puede ver en la figura 32, los títulos de los vídeos que llegan a *trending* suelen describir hechos y expresan un sentimiento en su mayoría neutro.
- ¿Podemos modelar en número de veces que un vídeo se ve y establecer un modelo predictivo?
- Si, aunque un estudio en más detalle sería necesario

Hemos obtenido múltiples conclusiones muy interesantes, hemos visto cómo los diferentes países interactúan con los videos, ya sea desde el punto de vista de como reaccionan a ellos o cuáles son los más populares.

Todas estas conclusiones serán de gran utilidad para cualquier creador de contenido que trate de hacerse un hueco en YouTube, o incluso grandes empresas que quieran realizar propuestas de marketing a través de la plataforma podrían utilizar ésta información para especializar los anuncios por comunidad o mercado.

Referencias

- [1] Leonard Berzkowitz, *Advances in Experimental Social Psychology*, 1967, Academic press
- [2] Richard McElreath, Robert Boyd, *Mathematical Models of Social Evolution: A Guide for the Perplexed*, 2007, The University of Chicago Press
- [3] Thomas L. Saaty and Joyce M. Alexander, *Thinking with models: Mathematical Models in the Physical, Biological, and Social Sciences*, 2015, RWS Publications
- [4] Jason Radford and Kenneth Joseph, *Theory In, Theory Out: The Uses of Social Theory in Machine Learning for Social Science*, 2020, doi.org/10.3389/fdata.2020.00018