Sentiment-Driven Video Recommendations

Iván Seldas Perulero

🎯 Sentiment-Driven Video Recommendations 🎯

Personalized video recommendation system based on **video content**, **user interactions**, and **sentiment analysis** from comments to recommend relevant videos to users.

📹 Input:

video_id	title
Z5s4cWbZX6E	The Ethics of Artificial Intelligence

🔝 Top 5 Recommendations

video_id	title	final_score
Aof4BxK0UlY	Artificial Intelligence Advances, and the Ethical Choices Ahead	1.36436
kX4oTF-2_kM	#12np: Artificial Intelligence is Hard to See: Social & ethical impacts of AI	1.30757
7Azhgh0nhBY	Artificial Intelligence: How It Will Impact the Financial Industry	1.2211
AT8JCkJH9pY	The Future of Artificial Intelligence - Shaping our AI Futures	1.1094
yIRL4xtmXE4	How Will Artificial Intelligence Change Ethics? - Pedro Domingos	1.10072

PROCESO


Video_id
01
02
03
Videos recomendados
MATRIZ DE SIMILITUD
ANÁLISIS DE SENTIMIENTOS
CLUSTERING

OBTENCIÓN DE DATOS REALES

YOUTUBE API SEARCH

TOPIC: Inteligencia Artificial

```
queries = [
    "What is artificial intelligence?" ,
    "Artificial intelligence applications in healthcare" ,
    "AI in autonomous vehicles" ,
    "Machine learning vs deep learning" ,
    "Artificial intelligence in finance" ,
    "How does AI work?" ,
    "Top AI tools for data science" ,
    "Artificial intelligence in robotics" ,
    "AI-driven innovation in business" ,
]
```

```
params = {
    'part': 'snippet',
    'type': 'video',
    'maxResults': 50,
    'key': api_key,
    'order': 'viewCount',
    'videoDuration': 'any',
    'regionCode': 'US'
}
```

DATAFRAMES

Canales

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2010 entries, 0 to 2009
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   channel_id        2010 non-null   object
 1   title             2010 non-null   object
 2   description       1857 non-null   object
 3   published_at      2010 non-null   object
 4   subscriber_count  2010 non-null   int64
 5   video_count       2010 non-null   int64
 6   view_count        2010 non-null   int64
 7   region            1583 non-null   object
dtypes: int64(3), object(5)
memory usage: 125.8+ KB
```

Videos

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2711 entries, 0 to 2710
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   videoId         2711 non-null   object
 1   title           2711 non-null   object
 2   channelId       2711 non-null   object
 3   description     2535 non-null   object
 4   publishedAt     2711 non-null   object
 5   thumbnail_url   2711 non-null   object
 6   tags            2711 non-null   object
 7   live_broadcast  2711 non-null   object
 8   categoryId      2711 non-null   int64
 9   viewCount       2711 non-null   int64
 10  likeCount       2711 non-null   int64
 11  commentCount    2711 non-null   int64
 12  licensed        2711 non-null   bool
 13  duration        2711 non-null   object
 14  caption         2711 non-null   bool
 15  language        2711 non-null   object
dtypes: bool(2), int64(4), object(10)
memory usage: 301.9+ KB
```

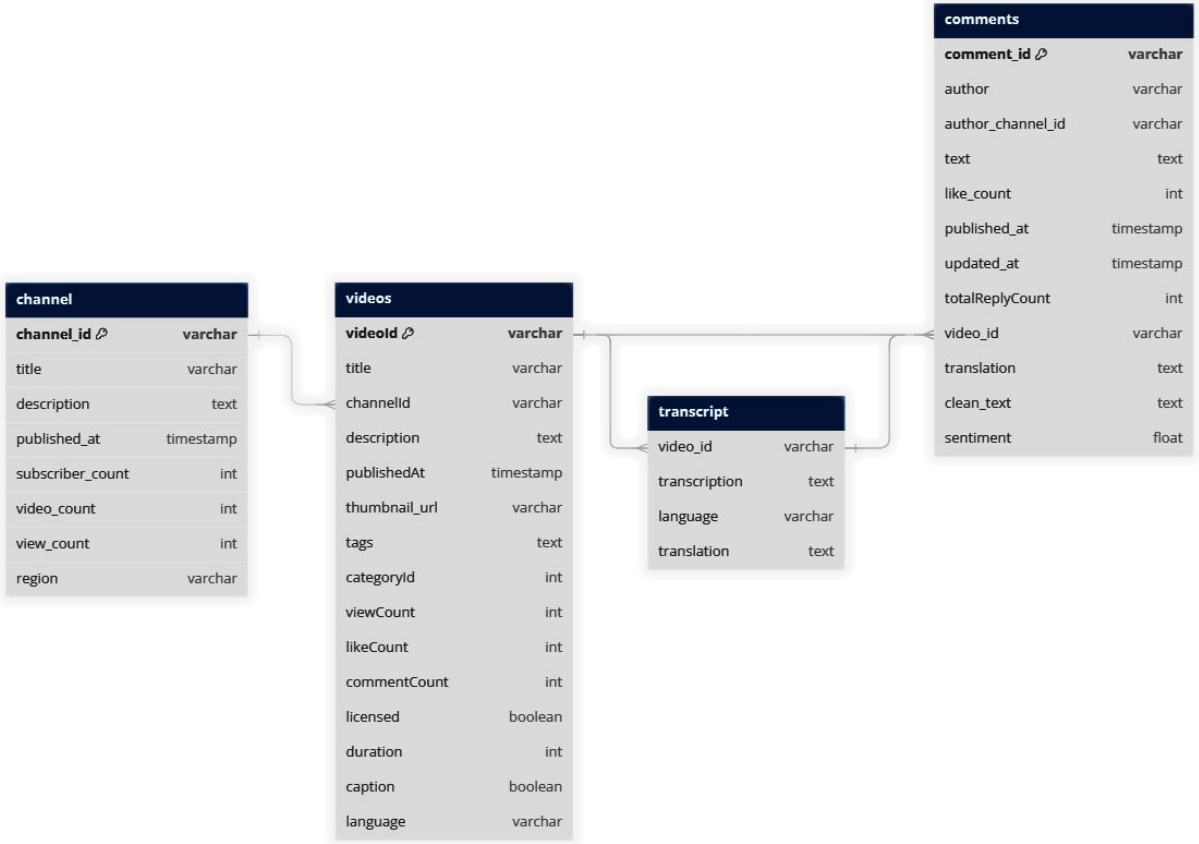
Transcripciones

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1787 entries, 0 to 1786
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   video_id       1787 non-null   object
 1   transcription  1787 non-null   object
 2   language       1787 non-null   object
dtypes: object(3)
memory usage: 42.0+ KB
```

Comentarios

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 185382 entries, 0 to 185381
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   comment_id         185382 non-null  object
 1   author             185003 non-null  object
 2   author_channel_id  185109 non-null  object
 3   text               185109 non-null  object
 4   like_count         184944 non-null  float64
 5   published_at       184944 non-null  object
 6   updated_at         184779 non-null  object
 7   totalReplyCount    184779 non-null  float64
 8   video_id           184779 non-null  object
dtypes: float64(2), object(7)
memory usage: 12.7+ MB
```

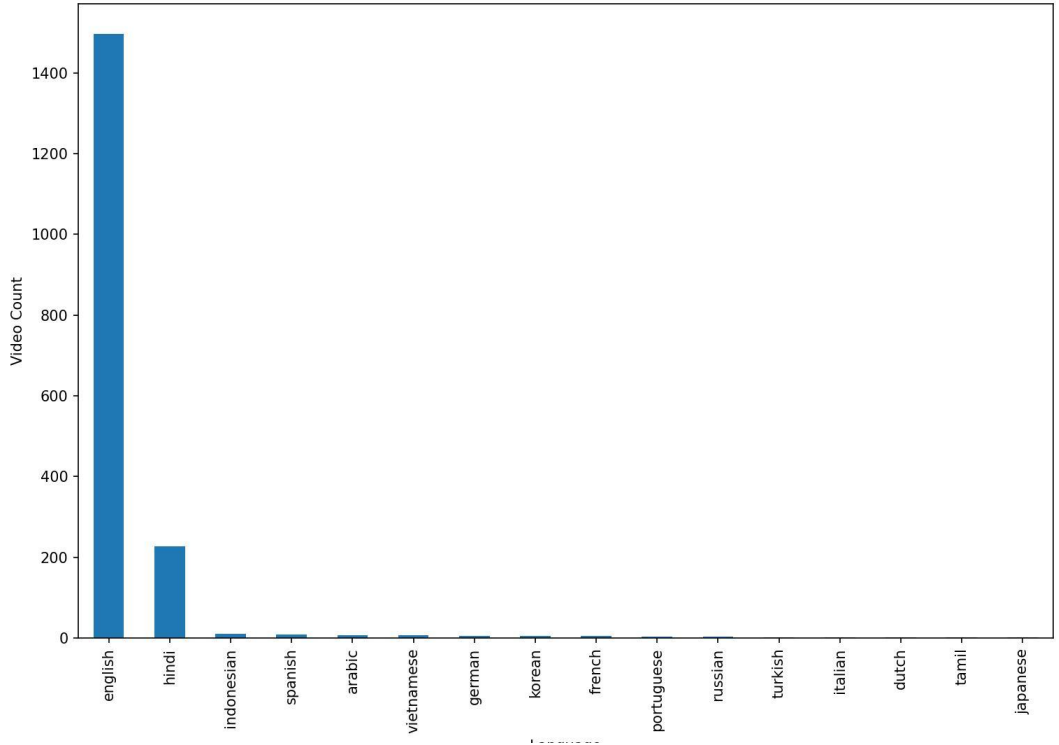
DATABASE


channel
channel_id varchar
title varchar
description text
published_at timestamp
subscriber_count int
video_count int
view_count int
region varchar
videos
videoId varchar
title varchar
channelId varchar
description text
publishedAt timestamp
thumbnail_url varchar
tags text
categoryId int
viewCount int
likeCount int
commentCount int
licensed boolean
duration int
caption boolean
language varchar
transcript
video_id varchar
transcription text
language varchar
translation text
comments
comment_id varchar
author varchar
author_channel_id varchar
text text
like_count int
published_at timestamp
updated_at timestamp
totalReplyCount int
video_id varchar
translation text
clean_text text
sentiment float

MATRIZ DE SIMILITUD

TRADUCCIÓN DE TRANSCRIPCIONES

	video_id	original_language	transcription	language
0	qtlUwwtvuEg	English (auto-generated)	[Music] thank you hello everyone I hope you ar...	english
1	QaoDXYYtgK0	English (auto-generated)	number three [Music] Facebook has enacted an e...	english
2	PqDwddEHswU	English (auto-generated)	in this series we're going to introduce deep l...	english
3	B-Y7rnOa43w	English (auto-generated)	this is how to earn money with AI and it's par...	english
4	vyit-1zKsZ4	English (auto-generated)	when current Medical Science has run out of op...	english



PREPROCESAMIENTO DEL TEXTO

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string
import re

# Descargar recursos de NLTK si no están descargados
nltk.download('wordnet')
nltk.download('stopwords')

# Inicializar el lematizador en inglés
lemmatizer = WordNetLemmatizer()

# Cargar las stopwords en inglés
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    if text is None:
        return None
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    text = text.translate(str.maketrans('', '',
string.punctuation))
    text = ' '.join([lemmatizer.lemmatize(word) for word in
text.split() if word not in stop_words])
    return text
```

- **Limpieza de caracteres**: Se eliminan caracteres no alfabéticos.
- **Normalización**: Convierte el texto a minúsculas.
- **Eliminación de ruido**: Se eliminan las puntuaciones y las stopwords.
- **Lematización**: Se reduce cada palabra a su forma básica o lema.

TF-IDF: Term Frequency - Inverse Document Frequency

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Donde:

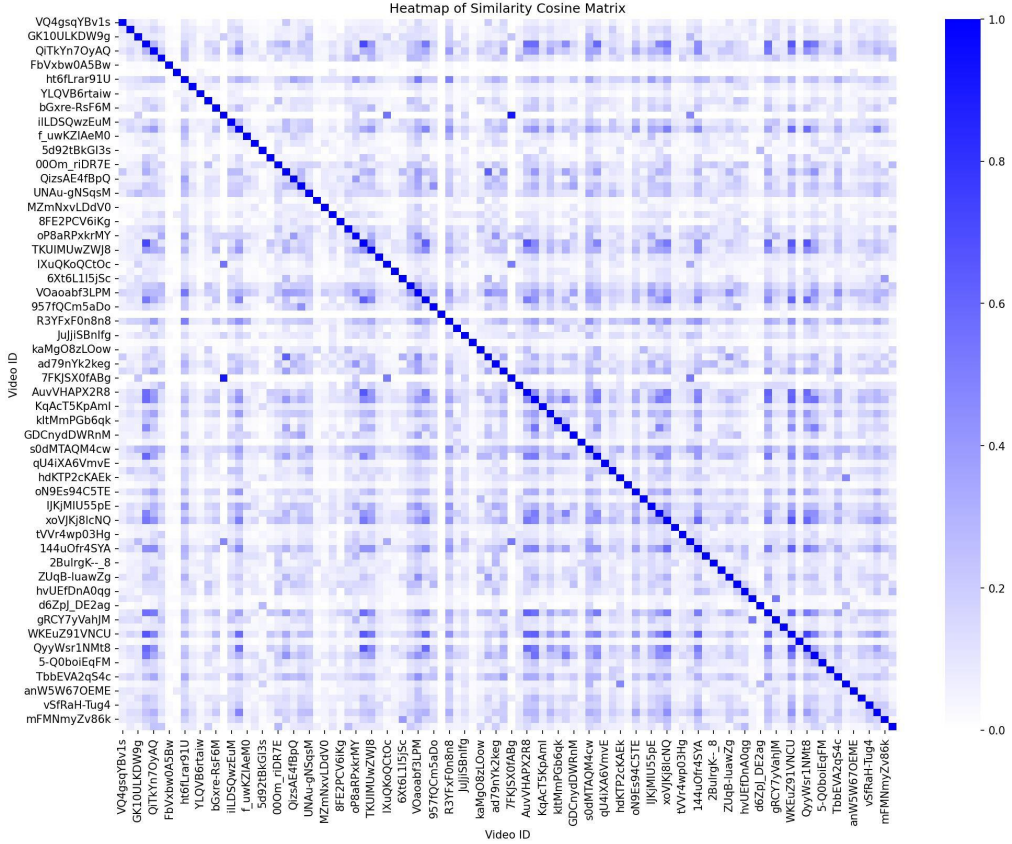
- **TF (Term Frequency)**: Mide la frecuencia con la que un término t aparece en un documento d. Se calcula como:

$$\text{TF}(t, d) = \frac{\text{Número de veces que el término } t \text{ aparece en el documento } d}{\text{Número total de términos en el documento } d}$$

- **IDF (Inverse Document Frequency)**: Mide la importancia de un término en todo el conjunto de documentos D. Se calcula como:

$$\text{IDF}(t, D) = \log\left(\frac{\text{Número total de documentos en el conjunto } D}{\text{Número de documentos que contienen el término } t}\right)$$

MATRIZ DE SIMILITUD: MAPA DE CALOR

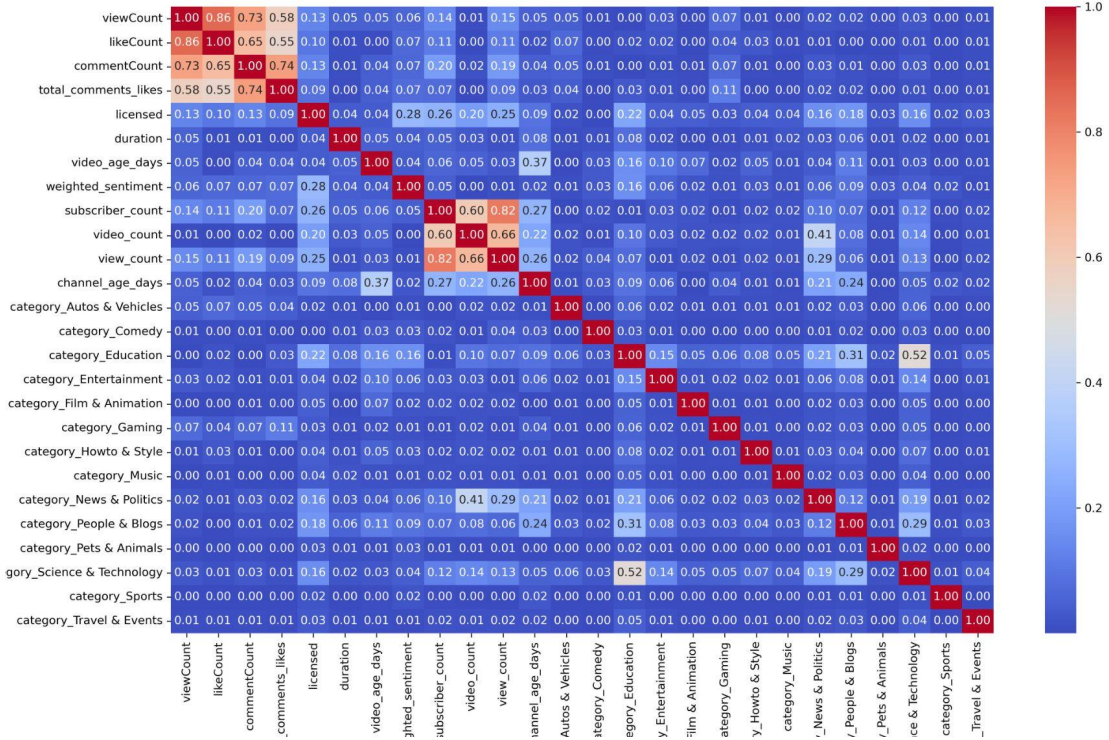


ANÁLISIS DE SENTIMIENTOS

NUBE DE PALABRAS

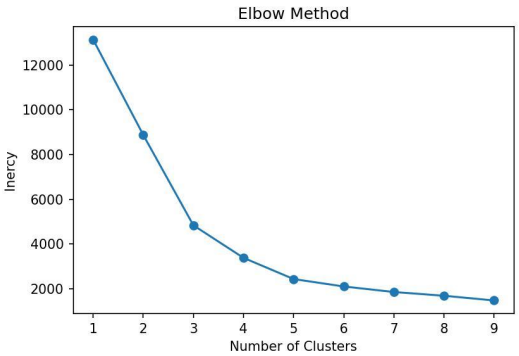
CLUSTERING

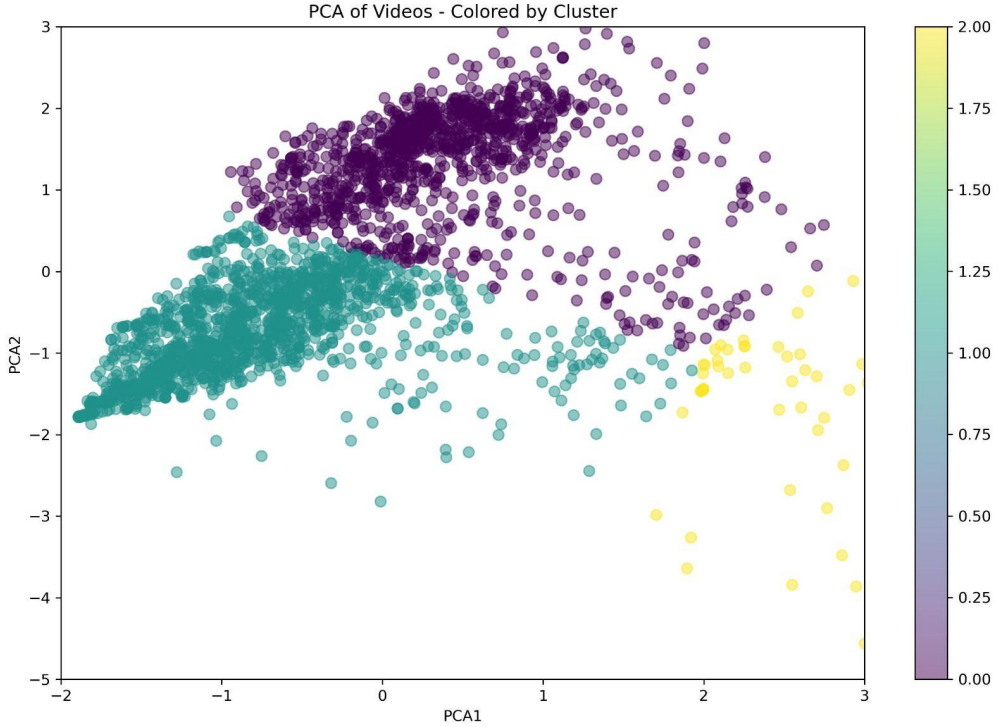
MATRIZ DE CORRELACIÓN



K-MEANS

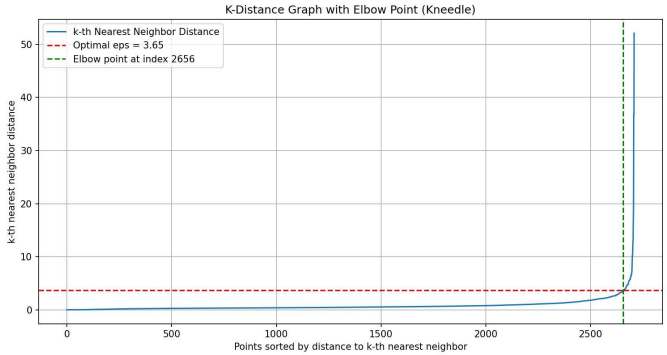
```
kmeans = KMeans(n_clusters=3)

Calculate the silhouette score
silhouette_avg_score = 0.4566
```

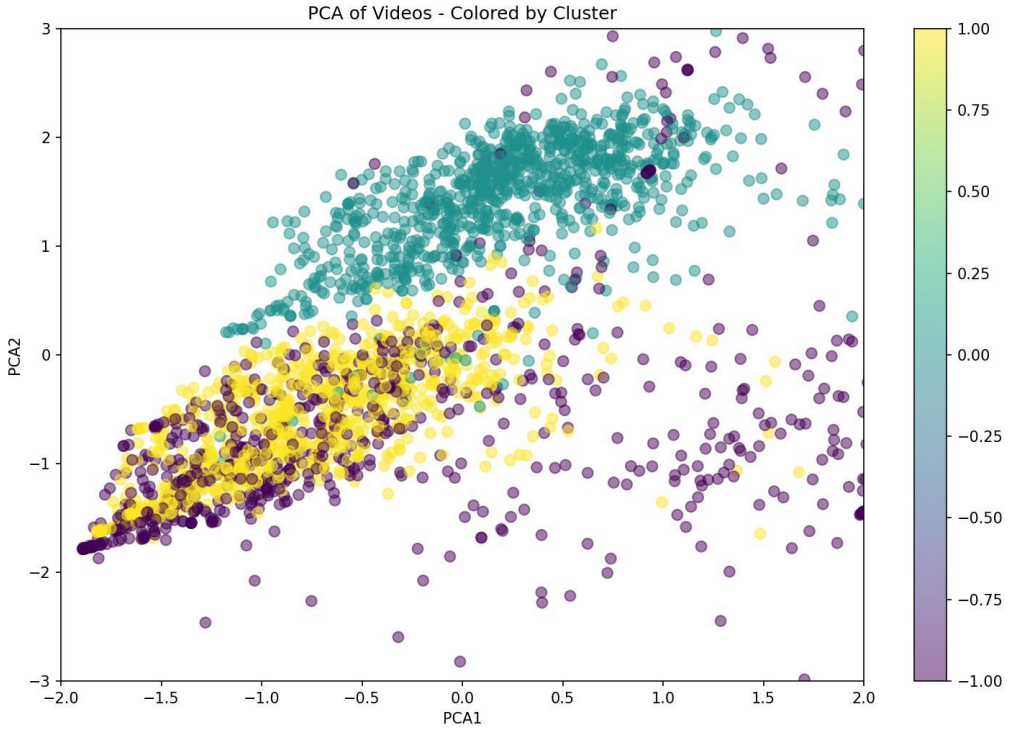




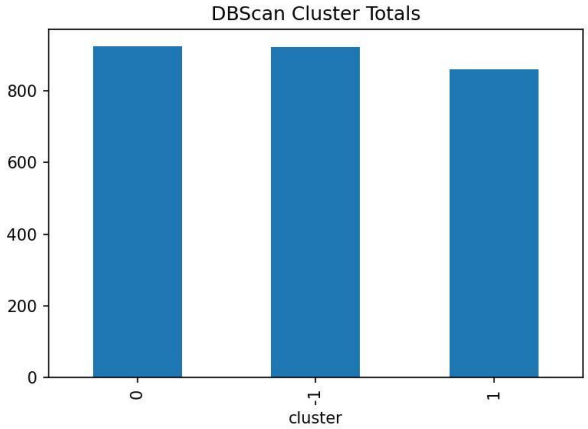
DBSCAN

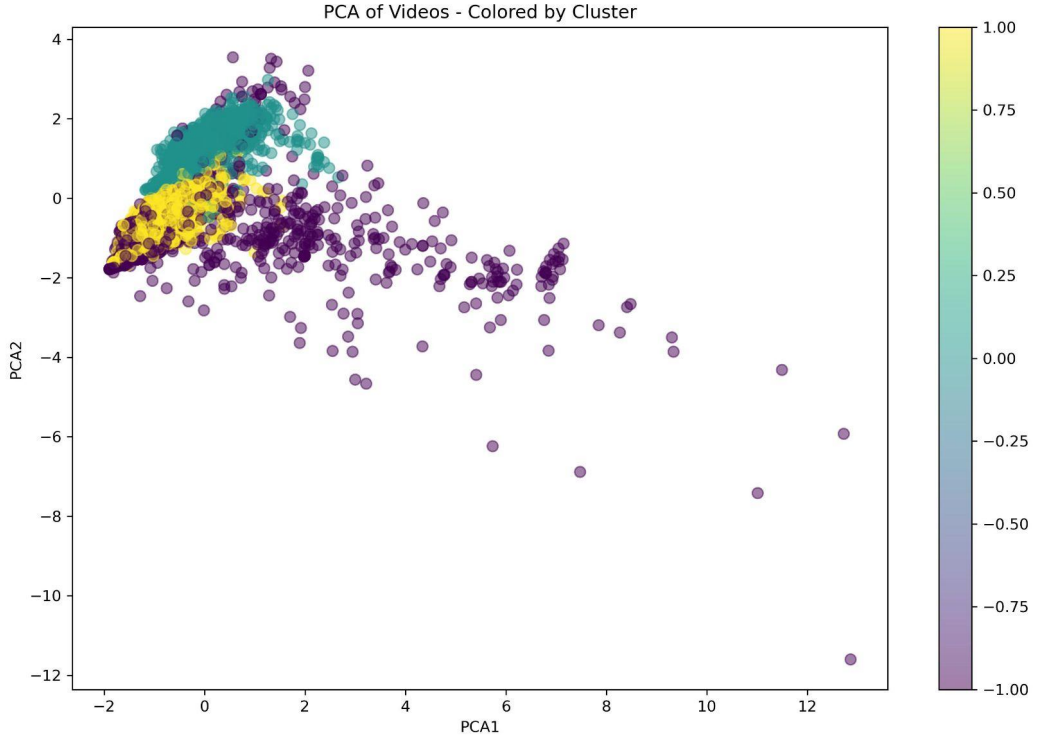
```
dbscan_model = DBSCAN(eps=2.65,
min_samples=450)

Number of clusters: 2
Number of noise points (-1): 762
silhouette_avg_score = 0.3891
```





CLUSTERIZACIÓN ELEGIDA: DBSCAN





APLICACIÓN FINAL

CÁLCULO DEL FINAL_SCORE

Final Score = TF-IDF x Weighted Sentiment × Clusters

PRÓXIMOS PASOS