Youtube Analyzer
Content, Sentiment and Performance Analysis Tool

Изработил:

Дејан Илоски 213074

https://github.com/dejo32/youtubeanalyzer

Ментор: Милена Трајаноска

Семинарска по предметот Вовед во наука за податоци тема 35

YouTube Channel Data Analysis: A Comprehensive Documentation of the YouTube Analyzer Tool

1. Introduction

In the dynamic landscape of online content creation, particularly on platforms like YouTube, understanding performance through data is no longer a luxury but a necessity. Content creators, marketers, and data enthusiasts increasingly rely on analytical insights to refine their strategies, optimize content, and foster audience engagement. Metrics such as view counts and engagement rates offer quantitative measures of reach, but a deeper understanding requires delving into qualitative aspects, such as audience sentiment and the underlying themes of successful content.

The youtubeanalyzer.txt script is a Python-based tool meticulously designed to automate the process of fetching, analyzing, and visualizing YouTube channel data. Its primary purpose is to empower users with a clear, data-driven perspective on channel performance. The tool's core functionalities are multifaceted: it efficiently retrieves video metadata, including titles, view counts, likes, comments, and durations, leveraging the robust YouTube Data API. Beyond raw metrics, it employs advanced natural language processing (NLP) techniques. Specifically, it performs sentiment analysis on video comments and titles, classifying them as Negative, Neutral, or Positive using a sophisticated pre-trained RoBERTa model. Furthermore, it conducts content analysis through Latent Dirichlet Allocation (LDA), identifying recurring themes and topics within the combined text of video titles and comments. The insights derived from these analyses are then transformed into intuitive visualizations, including various plots that represent performance metrics, sentiment distributions, and content trends. The key outputs generated by the tool include a comprehensive CSV file,

youtube_channel_analysis.csv, containing all processed data, alongside a series of insightful graphical representations. This comprehensive approach, which integrates both quantitative and qualitative data, moves beyond superficial metrics to offer a profound understanding of content performance, including the underlying reasons for success or failure and the audience's emotional response. This integration is vital for optimizing not just reach, but also the depth of audience engagement and positive reception, which are foundational for enduring channel growth.

This documentation is specifically tailored for YouTubers, content creators, marketers, and data enthusiasts who seek to harness the power of data for their channels. While a basic understanding of Python is assumed for users interacting with the script, complex concepts and technical details are explained with clarity and conciseness to ensure accessibility. The tool's existence and the documentation's focus on user accessibility are deliberate design choices, aiming to democratize sophisticated data analysis techniques.

2. Getting Started

To effectively utilize the YouTube Analyzer tool, a proper setup of the computational environment and dependencies is essential. This section outlines the prerequisites, library installations, API key configuration, and model loading procedures.

Prerequisites

To run the youtubeanalyzer.txt script, the following foundational requirements must be met:

- **Python 3.x:** A compatible Python environment is necessary. While the script does not specify a minimum version, Python 3.8 or newer is generally recommended for compatibility with modern libraries.
- **Internet Connection:** An active internet connection is required for fetching data via the YouTube Data API and for automatically downloading the sentiment analysis model and NLTK resources during execution.
- **Google Colab (Optional but Recommended):** Google Colab is highly recommended for running this script. Its cloud-based environment offers seamless integration with google.colab.userdata for secure API key management and often comes with many required libraries pre-installed, simplifying the setup process. While adaptable to other Python environments, users running the script locally would need to replace

google.colab.userdata with an alternative secure method for handling API keys, such as environment variables. The explicit mention of Google Colab and userdata in the script underscores a deliberate design choice to lower the barrier to entry for users who may not be familiar with intricate local environment setups or secure credential management

Installation of Required Libraries

The youtubeanalyzer.txt script relies on several Python libraries for its various functionalities. The primary library that typically requires manual installation via pip is transformers:

Bash

```
pip install transformers
```

Beyond transformers, the script imports a suite of other libraries that are crucial for its operation
:

- googleapiclient.discovery: Facilitates interaction with Google APIs, specifically the YouTube Data API.
- pandas: Essential for data manipulation and analysis, particularly for managing and structuring data in DataFrames.
- google.colab.userdata: Enables secure retrieval of user data, such as API keys, within a Google Colab environment.
- matplotlib.pyplot and seaborn: Core libraries for generating static and informative data visualizations.
- numpy: Provides fundamental numerical operations, crucial for array manipulation and mathematical functions.
- time: Used to introduce delays in API requests, preventing quota errors.
- re: Python's built-in module for regular expressions, utilized in text preprocessing.
- googleapiclient.errors.HttpError: For handling HTTP-specific errors that may arise during API calls.

- transformers.AutoTokenizer and transformers.AutoModelForSequenceClassification: Components from the transformers library for loading pre-trained models and tokenizers for sentiment analysis.
- scipy.special.softmax: Computes the softmax function, converting raw model outputs into probabilities.
- collections.Counter: Used for counting hashable objects, particularly for determining dominant sentiments.
- sklearn.feature_extraction.text.TfidfVectorizer and sklearn.decomposition.LatentDirichletAllocation: Scikit-learn components for TF-IDF vectorization and Latent Dirichlet Allocation (LDA) in topic modeling.
- nltk: The Natural Language Toolkit, a comprehensive platform for working with human language data.
- nltk.corpus.stopwords and nltk.stem.WordNetLemmatizer: NLTK modules for accessing common stopwords and performing lemmatization.

The script also includes direct commands to download necessary NLTK resources, which will execute automatically upon running the script :

Python

```
import nltk

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('omw-1.4')
```

These downloads provide:

- 'stopwords': A list of common words (e.g., "the," "is," "and") often removed from text to focus on more meaningful terms.
- 'wordnet': A lexical database essential for the lemmatization process.
- 'omw-1.4': A dependency for WordNet, ensuring its proper functioning.

Setting Up YouTube API Key

To interact with the YouTube Data API, an API key is indispensable. The script is configured to retrieve this key securely, particularly within a Google Colab environment.

Users must first obtain a YouTube Data API key from the Google Cloud Console. This involves creating a new project (if one doesn't exist), enabling the YouTube Data API v3, and then generating API credentials. Once the API key is obtained, it should be stored securely. The script specifically uses userdata.get('YoutubeDataAPI'), which implies the key should be stored in Google Colab's userdata secrets manager.

To set up the API key in Google Colab:

1. Open the Colab notebook.

2. On the left sidebar, click the "🔑 Secrets" icon.
3. Click "Add new secret."
4. For "Name," enter YoutubeDataAPI.
5. For "Value," paste the actual YouTube Data API key.
6. Ensure "Notebook access" is toggled on for the specific notebook.

This method of retrieving the API key from userdata is a robust security practice, preventing the sensitive key from being hardcoded directly into the script

Loading the Sentiment Analysis Model

A significant component of the YouTube Analyzer is its sentiment analysis capability. The script automatically loads a pre-trained sentiment analysis model from Hugging Face during its initialization phase.

The model utilized is cardiffnlp/twitter-roberta-base-sentiment. The AutoTokenizer and AutoModelForSequenceClassification classes from the transformers library are employed to load this model and its corresponding tokenizer. This global loading ensures that the model is ready for use throughout the script's execution without requiring repeated loading, which would be inefficient. The decision to leverage a pre-trained RoBERTa model offers substantial advantages: it bypasses the need for extensive data collection and training from scratch, thereby conserving significant computational resources and time.

3. Code Structure and Functionality

This section provides a detailed examination of the youtubeanalyzer.txt script's architecture, outlining its workflow and explaining the purpose and operation of each key function. The structured approach ensures data integrity and efficient processing, with embedded resource management critical for a tool relying on external APIs.

Overview of the Code Flow

The youtubeanalyzer.txt script follows a logical, sequential workflow designed to systematically fetch, analyze, visualize, and save YouTube channel data. This structured progression ensures that each stage of processing builds upon the successful completion of the preceding one, maintaining data integrity and facilitating efficient resource utilization.

1. **Initialization and Setup:** The script begins by importing all necessary libraries, including those for API interaction, data manipulation, NLP, and visualization. It then proceeds to download essential NLTK resources (stopwords, wordnet, omw-1.4). Following this, the YouTube Data API is initialized using a securely retrieved API key, and the pre-trained sentiment analysis model (

cardiffnlp/twitter-roberta-base-sentiment) is loaded globally. Finally, NLP tools such as stopwords and a lemmatizer are prepared for text processing.

2. **Data Fetching:** The core data acquisition phase involves retrieving the specified channel's uploads playlist ID. Subsequently, video IDs are fetched from this playlist, often in an iterative manner with pagination. Detailed information for these videos (e.g., titles, statistics, content details) is then retrieved in batches. A crucial aspect of this stage is the

limited fetching of top comments, which is performed only for the first 10 videos to manage API quota usage.

3. **Analysis:** With the raw data in hand, the script moves to the analytical phase. Sentiment analysis is performed on both the collected comments and the video titles. Concurrently, content analysis, specifically topic modeling using Latent Dirichlet Allocation (LDA), is conducted on the combined and preprocessed text from video titles and comments.
4. **Visualization:** The insights derived from the analysis are then transformed into a series of visual representations. Various plotting functions are called to generate charts that illustrate performance metrics, sentiment distributions, and identified content topics.
5. **Output and Saving:** The final step involves cleaning the processed DataFrame by removing temporary columns and then saving the comprehensive results to a CSV file named youtube_channel_analysis.csv.

Utility Functions

The youtubeanalyzer.txt script incorporates several utility functions that serve as foundational pillars for its more complex operations, particularly in handling API requests efficiently and preparing text data for robust NLP tasks.

- **chunks(lst, n):**
  - **Functionality:** This function operates as a generator, designed to yield successive n-sized batches from a given list (lst). It iterates through the list, producing sub-lists of a specified size in each step. The use of

yield makes it memory-efficient, as it generates items on demand rather than creating an entire list of chunks in memory simultaneously.

  - **Purpose:** This utility is crucial for managing API requests effectively. For instance, in the get_video_data function, it breaks down large lists of video IDs into smaller, manageable batches (e.g., 50 video IDs per request for videos().list). This directly aids in adhering to the YouTube Data API's

maxResults limits, which specify the maximum number of items that can be retrieved in a single call, thereby preventing quota errors and ensuring more reliable data retrieval.

- **preprocess_text(text):**
  - **Functionality:** This function is responsible for cleaning and normalizing raw text data, making it suitable for subsequent natural language processing (NLP) tasks. The process involves several steps: converting all text to lowercase, removing non-alphabetic characters and punctuation using regular expressions, filtering out common English stopwords (e.g., "the," "is," "and"), and lemmatizing the remaining words. Lemmatization reduces words to their base or root form (e.g., "running," "ran," "runs" all become "run"), which is critical for accurate text analysis. Additionally, it filters out very short words (less than 3 characters) that are often noise or less informative.
  - **Purpose:** This function is essential for preparing raw text (from video titles and comments) for accurate NLP tasks like topic modeling. By normalizing the text, it

ensures that variations of the same word are treated as a single entity, significantly improving the quality and consistency of topic extraction and other text-based analyses.

- **create_topic_name(keywords):**
    - **Functionality:** This utility takes a comma-separated string of keywords, typically derived from topic modeling, and generates a human-readable topic name. It selects the first three keywords from the input string, capitalizes each, and then joins them with spaces to form a concise and descriptive topic label.
    - **Purpose:** This function significantly enhances the interpretability of topic modeling results for the end-user. Instead of abstract topic IDs or a long list of keywords, users are presented with meaningful names like "Gaming Challenges" or "Product Reviews." This transformation from technical output to user-friendly insights makes the analysis results more immediately actionable and understandable for content creators and marketers.

These utility functions, while seemingly simple, are fundamental to the analyzer's overall reliability and value. They address both technical constraints, such as API limits, and user experience considerations, by providing interpretable results.

Fetching Video Data (get_video_data)

The get_video_data function is central to the YouTube Analyzer, responsible for acquiring comprehensive video data from a specified YouTube channel. Its design incorporates robust mechanisms for data retrieval, performance metric calculation, and strategic API quota management.

The process begins by identifying the channel's uploads playlist ID through a youtube.channels().list API request. This playlist serves as the source for all public videos uploaded by the channel. Subsequently, the function iteratively fetches video IDs from this playlist using

youtube.playlistItems().list. This process handles pagination via nextPageToken to ensure all available videos are considered, up to a specified max_videos limit (e.g., 50 videos in the main execution block). This limit is a critical component of the tool's API quota management strategy, preventing excessive requests.

Once the video IDs are collected, the function proceeds to fetch detailed information for these videos. It uses youtube.videos().list to retrieve comprehensive details, including snippets (title, published date), statistics (views, likes, comments), and content details (duration). To optimize API usage and adhere to rate limits, these requests are made in batches (chunks of 50 video IDs), with a

time.sleep(1) pause after each batch to prevent hitting quota limits too quickly.

A strategic decision in get_video_data involves the processing of comments. To conserve API quota, get_top_comments is called to fetch comments *only for the first 10 videos* in the fetched list. This approach ensures that some qualitative sentiment data is collected while significantly reducing the number of quota-intensive comment thread requests. The function also includes error handling for disabled comments, silently skipping such videos to maintain script continuity.

For each video, a suite of performance metrics is calculated:

- **Views:** The total number of times the video has been watched.
- **Likes/Dislikes:** The count of positive and negative reactions.
- **Comment Count:** The total number of comments received.
- **Engagement Rate:** Calculated as (likes + dislikes + comment_count) / views * 100, this metric provides a normalized measure of audience interaction relative to viewership.
- **Like Ratio:** Determined by likes / (likes + dislikes) * 100, indicating the percentage of positive reactions among total reactions.
- **Is Short:** A boolean flag indicating if the video is a YouTube Short, based on its duration (less than 60 seconds and no 'M' for minutes).

Robust try-except blocks are integrated throughout the get_video_data function to catch and manage various API errors and processing exceptions. After all data is collected, the function performs post-processing steps: it adds sentiment analysis results for video titles and creates a

processed_content column by combining titles and preprocessed top comments. This combined text is then prepared for subsequent topic modeling. The strategic balance between providing comprehensive insights and managing API costs/limitations means that comment sentiment is a sample-based understanding, not a comprehensive one across all fetched videos. This distinction is important for users interpreting overall channel sentiment, especially for very large channels.

Sentiment Analysis (analyze_text_sentiment)

The analyze_text_sentiment function is responsible for assessing the emotional tone of text, specifically video comments and titles, using a sophisticated pre-trained model. This capability provides qualitative insights that complement the quantitative metrics of channel performance.

The function takes a list of texts as input and processes them efficiently in batches, with a batch_size set to 32. For each batch, the

tokenizer (initialized with the cardiffnlp/twitter-roberta-base-sentiment model) converts the raw text into numerical inputs suitable for the model. This tokenization process includes padding shorter sequences and truncating longer ones to a max_length of 128 tokens, ensuring uniform input dimensions for the model.

These tokenized inputs are then fed into the model (an AutoModelForSequenceClassification instance of the same RoBERTa model). The model generates logits, which are raw, unnormalized scores for each sentiment class (Negative, Neutral, Positive). To transform these logits into interpretable probabilities, the

softmax function from scipy.special is applied. This converts the scores into a probability distribution where each value represents the likelihood of the text belonging to a specific sentiment class, with the sum of probabilities for each text equaling 1.

For each probability distribution, the function identifies the label_idx corresponding to the highest probability, which represents the predicted sentiment. The sentiment label (from ['Negative', 'Neutral', 'Positive']) and its associated confidence score (the probability of the predicted label) are then appended to respective lists. The use of a pre-trained RoBERTa model in this function exemplifies leveraging transfer learning, allowing the tool to benefit from state-of-the-art NLP capabilities without requiring extensive data or training from the user.

Content Analysis (perform_content_analysis)

The perform_content_analysis function is designed to uncover the underlying themes and topics within a YouTube channel's video content, providing a deeper understanding of content strategy and audience interest. This process involves several critical steps, from text preparation to topic identification and naming.

The function first ensures that sufficient processed_content (a combination of video titles and top comments after initial cleaning) is available for meaningful analysis. It filters out entries with less than 20 characters and requires at least 5 valid content entries to proceed, preventing analysis on sparse data.

The core of the analysis begins with **TF-IDF Vectorization**. The TfidfVectorizer is initialized with max_features=1000 to limit the vocabulary to the most important terms and ngram_range=(1, 2) to consider both single words (unigrams) and two-word phrases (bigrams). This vectorizer transforms the preprocessed text into a numerical matrix, where each row represents a video and each column a unique term, with values indicating the term's importance within that video relative to the entire corpus.

Next, **Latent Dirichlet Allocation (LDA) for Topic Modeling** is applied. The number of topics (n_topics) is dynamically determined, capped at a maximum of 5, but ensuring at least one less than the number of valid documents to avoid errors with small datasets. An

LatentDirichletAllocation model is trained on the TF-IDF vectors, identifying latent topics and assigning a dominant_topic to each video based on its topic distribution.

To make these identified topics interpretable, the function extracts the most representative keywords for each topic from the LDA components. These keywords are then passed to the create_topic_name utility function, which generates a human-readable name for each topic (e.g., "Gaming Challenges" from keywords like "game, challenge, play"). This step is crucial for translating complex statistical outputs into actionable insights.

Finally, the dominant_topic and its topic_name are merged back into the main DataFrame, providing a categorized view of the channel's content. By linking these discovered topics to performance metrics (views, engagement), creators can pinpoint which content types are most successful, even if they hadn't explicitly defined those categories. This capability moves beyond simple performance tracking to offer strategic guidance, enabling creators to capitalize on successful content niches or re-evaluate underperforming areas, potentially revealing new content opportunities.

Visualizations

The YouTube Analyzer includes a suite of five dedicated plotting functions designed to transform complex analytical results into intuitive, easily interpretable graphical formats.

- **plot_top_videos(df):** Generates a horizontal bar chart displaying the top 10 most viewed videos. This visualization quickly highlights the channel's most popular content, allowing users to identify which video titles and themes have historically attracted the largest audience.
- **plot_comment_sentiment_distribution(df):** Presents the overall sentiment distribution of all collected comments through a pie chart and a bar chart. This provides a high-level overview of the general emotional tone of the audience's reactions across the analyzed videos, indicating whether comments are predominantly positive, neutral, or negative.
- **plot_video_sentiment_summary(df):** Creates a bar chart showing the count of videos for which a specific sentiment (Negative, Neutral, or Positive) was dominant in their comments. This offers a video-level perspective on audience reception, indicating how many individual pieces of content elicited a particular emotional response.
- **plot_content_performance(df):** This is a multi-plot visualization providing a comprehensive view of content topic performance. It includes:
    - Horizontal bar charts for average views and average engagement rate by content topic, showing which topics are most popular and interactive.
    - A horizontal bar chart for video distribution by content topic, illustrating the channel's content focus.
    - A "Performance Matrix" bubble chart, plotting average views against average engagement, with bubble size representing the number of videos in each topic. This matrix is particularly powerful for identifying "star" topics (high views, high engagement), niche but engaging content, or underperforming areas.
- **plot_performance_metrics(df):** Offers a deeper statistical examination of various performance indicators through a set of four plots:
    - A box plot combined with a swarm plot illustrating the distribution of video views (on a logarithmic scale to handle skewed data).
    - A violin plot showing the distribution of engagement rates.
    - A histogram with a Kernel Density Estimate (KDE) for the calculated "performance score" (views multiplied by engagement rate).
    - A scatter plot with a regression line depicting the relationship between engagement rate and views (also on a logarithmic scale), accompanied by the correlation coefficient to quantify their relationship.

4. Using the Analyzer

This section provides practical guidance on executing the youtubeanalyzer.txt script and interpreting its outputs, including the detailed CSV file and the various graphical visualizations.

Running the Code

Executing the YouTube Analyzer tool involves a few straightforward steps:

1. **Specify Channel ID:** The first step is to identify the YouTube channel intended for analysis. In the MAIN EXECUTION block of the script, locate the CHANNEL_ID variable:

Python

```
CHANNEL_ID = "UCX6OQ3DkcsbYNE6H8uQQuVA"     # MrBeast channel
```

Replace "UCX6OQ3DkcsbYNE6H8uQQuVA" with the unique ID of the target YouTube channel. This ID can typically be found in the channel's URL (e.g., youtube.com/channel/CHANNEL_ID) or by using online tools designed to extract channel IDs.

2. **Execute Data Fetching:** The primary function call to initiate data retrieval is get_video_data. This function fetches video metadata and comments from the specified channel. The max_videos parameter can be adjusted to control the number of videos retrieved, which is a crucial consideration for managing YouTube Data API quota usage. For example, to fetch data for up to 50 videos:

Python

```
df = get_video_data(CHANNEL_ID, max_videos=50)
```

3. **Run Analysis and Visualization Functions:** Once the data is fetched and stored in the df DataFrame, the script proceeds to perform content analysis and generate visualizations. These functions are called sequentially:

Python

```
if not df.empty:
    df, topic_names = perform_content_analysis(df)
    #... (print topic names, sample data)...
    plot_top_videos(df)
    plot_comment_sentiment_distribution(df)
    plot_video_sentiment_summary(df)
    if 'topic_name' in df.columns and not df['topic_name'].isnull().all():
        plot_content_performance(df)
    plot_performance_metrics(df)
    #... (save results)...
```

Simply running the entire script will execute these functions in the correct order, displaying progress messages in the console (e.g., "YouTube API initialized," "Sentiment model loaded," "Fetching data," "Content Topics Identified," "Results saved").

Interpreting Results

The YouTube Analyzer provides its results in two primary formats: a detailed CSV file and a series of informative plots.

CSV Output (youtube_channel_analysis.csv)

The youtube_channel_analysis.csv file serves as a comprehensive tabular summary of the analysis, providing granular data for each video. Each row in the CSV corresponds to a single video, with columns detailing various metrics and analytical insights.

Table 1: youtube_channel_analysis.csv Output Columns

Column Name	Description
video_id	Unique identifier assigned by YouTube to each video.
title	The official title of the YouTube video.
published_at	The date and time when the video was published on YouTube.
views	The total number of times the video has been viewed.
likes	The total number of 'likes' the video has received.
dislikes	The total number of 'dislikes' the video has received.
comments	The total number of comments posted on the video.
duration	The video's length, typically in ISO 8601 duration format (e.g., PT10M30S for 10 minutes and 30 seconds).
is_short	A boolean (True/False) indicating if the video is classified as a YouTube Short (duration less than 60 seconds and no 'M' for minutes).
engagement_rate	A calculated metric: (likes + dislikes + comment_count) / views * 100. Indicates audience interaction relative to viewership; higher percentages suggest better engagement.
like_ratio	A calculated metric: likes / (likes + dislikes) * 100. Represents the percentage of positive reactions among total reactions; higher ratios indicate better reception.
top_comments	A list of the top comments fetched for the video (note: comments are only fetched for the first 10 videos to manage API quota).
dominant_comment_sentiment	The most frequently occurring sentiment (Negative, Neutral, or

Column Name	Description
	Positive) identified from the top_comments for that video.
title_sentiment	The sentiment (Negative, Neutral, or Positive) of the video's title, as analyzed by the pre-trained sentiment model.
dominant_topic	The numerical ID assigned to the primary content topic identified for the video through Latent Dirichlet Allocation (LDA).
topic_name	A human-readable name for the dominant_topic, generated from its most representative keywords.

Export to Sheets

Interpreting the CSV data involves examining these columns. For instance, a video with a high views count but a comparatively low engagement_rate might suggest that while the content attracted initial attention, it did not strongly resonate with the audience or provoke significant interaction. Conversely, a video with moderate views but a high engagement_rate could indicate a highly engaged niche audience. The dominant_comment_sentiment column provides a quick overview of the general emotional tone of the discussion surrounding a video, which is crucial for understanding audience reception beyond just simple like/dislike counts.

Visualizations

The script generates several plots, each designed to offer specific insights into channel performance and content trends. These visualizations transform complex numerical and textual data into intuitive narratives, enabling content creators to quickly identify strengths, weaknesses, and opportunities for their content strategy without needing to delve into raw data tables.
plot_top_videos(df): Top 10 Videos by Views

- **Interpretation:** This horizontal bar chart clearly identifies the channel's most popular videos based on their view counts. It helps in understanding which content themes, formats, or specific videos have historically attracted the largest audience. Consistent appearance of a particular content type here indicates a successful content pillar.

2. **plot_comment_sentiment_distribution(df): Overall Comment Sentiment & Sentiment Distribution**

   - **Interpretation:** This dual plot (pie chart and bar chart) provides a high-level overview of the collective sentiment across all analyzed comments. The pie chart offers a quick visual breakdown of Negative, Neutral, and Positive percentages, while the bar chart allows for easier comparison of magnitudes. For example, if the pie chart shows a predominant percentage of "Positive" comments, it indicates a generally supportive and appreciative community.

3. **plot_video_sentiment_summary(df): Dominant Sentiment per Video**

   - **Interpretation:** This bar chart illustrates the number of videos whose comments predominantly fall into Negative, Neutral, or Positive sentiment categories. It helps to understand the sentiment trend at a video-specific level, revealing if most content pieces are consistently well-received or if certain videos elicit stronger negative reactions.

4. **plot_content_performance(df): Performance Metrics by Content Topic**
    - **Interpretation:** This multi-plot is a cornerstone for strategic content planning.
        - **Average Views by Content Topic:** Horizontal bars show which content topics, on average, attract the most views.
        - **Average Engagement Rate by Content Topic:** Indicates which topics generate the most audience interaction relative to their views, suggesting content that deeply resonates.
        - **Video Distribution by Content Topic:** Displays the number of videos categorized under each topic, revealing the channel's content focus.
        - **Performance Matrix (Bubble Chart):** This scatter plot is particularly insightful. The X-axis represents average views, the Y-axis represents average engagement rate, and the size of each bubble corresponds to the number of videos within that topic.
            - Topics in the **top-right quadrant** (high views, high engagement, often large bubbles) represent "star" content—highly popular and deeply engaging, indicating areas of strong channel performance.
            - Topics in the **top-left quadrant** (low views, high engagement) might be niche content that deeply engages a smaller, dedicated audience.
            - Topics in the **bottom-right quadrant** (high views, low engagement) are popular but may not foster much interaction, suggesting opportunities to enhance engagement strategies.
            - Topics in the **bottom-left quadrant** (low views, low engagement) are underperforming and may require re-evaluation or discontinuation.
5. **plot_performance_metrics(df): Various Performance Metrics**
    - **Interpretation:** This set of four plots provides a deeper statistical understanding of key performance indicators.
        - **Video Views Distribution (Box Plot with Swarm Plot):** Illustrates the spread, central tendency, and outliers of video views. The logarithmic scale on the X-axis is crucial for visualizing the often skewed distribution of views.
        - **Engagement Rate Distribution (Violin Plot):** Shows the density and quartiles of engagement rates, providing insight into the typical range and variability of audience interaction.
        - **Performance Score Distribution (Histogram with KDE):** Displays the frequency distribution of a calculated performance_score (views multiplied by engagement rate, scaled), offering a combined measure of video success.

    - **Engagement vs Views (Scatter Plot with Regression Line):** Plots engagement rate against views (log scale) to identify any correlation. The displayed correlation coefficient quantifies this relationship (e.g., a coefficient of 0.70 indicates a strong positive correlation), suggesting that as views increase, engagement also tends to rise.

5. Advanced Topics

This section explores customization options for the YouTube Analyzer, strategies for managing YouTube Data API quotas, and common troubleshooting steps, providing users with the knowledge to optimize and maintain the tool's performance.

Customizing the Analyzer

The youtubeanalyzer.txt script offers several parameters that can be adjusted to tailor the analysis to specific needs, balancing data depth with API quota considerations. This adaptability allows the tool to cater to diverse analytical requirements and channel sizes.

- **max_videos:** This parameter in the get_video_data function controls the total number of videos fetched from a channel. Users can modify this value to retrieve a larger or smaller sample of videos, depending on the desired scope of analysis and the available API quota.
- **max_results for Comments:** Within the get_top_comments function, the max_results parameter (currently set to 10) determines the maximum number of top comments retrieved per video. Increasing this value will provide a more comprehensive sentiment analysis for individual videos but will also consume significantly more API quota.
- **Adding Metrics:** The get_video_data function can be extended to fetch additional metrics available through the YouTube Data API, such as subscriber counts (if accessible via channel statistics, though not directly in video statistics). Users can also calculate new derived metrics, such as custom engagement scores or watch time if that data becomes available through the API, to create more specialized insights.
- **LDA n_topics:** In the perform_content_analysis function, the n_topics parameter for LatentDirichletAllocation determines the number of content themes to be identified. This value, dynamically set to a maximum of 5, can be tweaked to explore more granular or broader content themes, aligning with the specific analytical goals.
- **TF-IDF Parameters:** Advanced users can fine-tune the text vectorization process by adjusting parameters within the TfidfVectorizer in perform_content_analysis. Specifically, max_features (currently 1000) controls the vocabulary size, and ngram_range (currently (1, 2) for unigrams and bigrams) can be modified to capture different linguistic patterns.

Handling Large Channels and API Quotas

Interacting with external APIs, such as the YouTube Data API, necessitates careful management of request quotas. The YouTube Data API imposes daily limits on the number of requests a project can make. The youtubeanalyzer.txt script incorporates several proactive strategies to mitigate the risk of exceeding these limits, ensuring the tool remains usable and cost-effective, particularly for users analyzing larger channels or performing frequent analyses.

- **Limiting Requests:** The most direct method implemented is setting a max_videos limit (e.g., 50 videos in the main execution) within the get_video_data function. Additionally, comment fetching is deliberately limited to only the first 10 videos. These explicit limitations significantly reduce the overall API calls made, directly managing quota consumption.
- **Batch Processing:** The script optimizes API calls by fetching video details in chunks of 50 video IDs rather than one by one. Similarly, sentiment analysis is performed on text in batches (e.g.,

batch_size=32). This batching reduces the overhead of individual requests and improves efficiency.

- **Rate Limiting:** A crucial strategy to prevent hitting rate limits too quickly is the inclusion of time.sleep(1) after each batch request for video details. This 1-second pause distributes requests over time, making it less likely to trigger rate-limiting mechanisms.
- **Caching Data:** An advanced strategy for users with large channels or frequent analysis needs is to implement data caching. Once data is fetched and saved to youtube_channel_analysis.csv, subsequent analysis or visualization runs can load data directly from this CSV file instead of re-fetching from the API. This significantly saves API quota for repeated operations.

These built-in safeguards for scalability and cost-effectiveness demonstrate a deep understanding of external API limitations. This proactive approach to quota management is essential for ensuring the tool's long-term usability and preventing unexpected service interruptions or charges.

Troubleshooting Common Issues

The youtubeanalyzer.txt script includes explicit error handling and informative messages designed to guide users through common pitfalls, thereby reducing the learning curve and improving the overall user experience.

- **API Key Errors:** If the script encounters issues with the YouTube Data API key, users should verify its setup. This involves ensuring the key is correctly obtained from the Google Cloud Console, the YouTube Data API v3 is enabled for the project, and the key is securely stored in Google Colab's userdata secrets manager (or configured as an environment variable if running locally).
- **Model Loading Failures:** Problems loading the sentiment analysis model typically stem from internet connectivity issues or missing/incorrect transformers library installations. Users should confirm a stable internet connection and ensure the transformers library is installed. If issues persist, verifying library versions might be necessary.
- **Empty Data or Insufficient Content:** The script provides specific messages when data is sparse or absent:

  - No videos found for this channel: This message indicates that the provided CHANNEL_ID might be incorrect, or the channel may not have any public videos available for fetching.
  - Insufficient content for topic modeling: This occurs if too few videos (specifically, fewer than 5 valid content entries) have sufficient text (titles combined with comments) for a meaningful topic analysis to be performed by LDA.
  - No comment sentiment data available: This message will appear if the channel has very few videos, or if comments are disabled on the first 10 videos (for which comments are fetched), resulting in no sentiment data for comments.
- **Quota Errors:** Despite built-in time.sleep mechanisms, intensive usage might still lead to HttpError messages related to API quota limits. If these errors occur, users can try increasing the time.sleep duration between batch requests or further reducing the max_videos parameter to lower the overall request volume.

6. Conclusion

The YouTube Analyzer tool represents a robust and accessible solution for content creators, marketers, and data enthusiasts seeking to understand and optimize their presence on YouTube. Its comprehensive capabilities integrate quantitative video metrics with qualitative insights, providing a holistic view of channel performance.

The tool's core strength lies in its ability to deliver detailed video metrics (views, engagement, likes, comments) and combine them with sophisticated natural language processing. It performs sentiment analysis on video comments and titles, offering a nuanced understanding of audience reactions, and utilizes topic modeling to identify recurring content themes. These analytical results are then transformed into a variety of intuitive visualizations, making complex data easily digestible and actionable. The analyzer's design also incorporates robust mechanisms for handling YouTube Data API interactions, including strategic quota management and error handling, ensuring reliable operation

The potential applications of the YouTube Analyzer are diverse and impactful:

- **Optimizing Content Strategy:** By identifying high-performing topics and formats, creators can understand what truly resonates with their audience and focus their efforts on producing more of what works.
- **Audience Understanding:** The sentiment analysis capabilities provide invaluable insights into the emotional reactions of the audience to specific videos or content types, allowing creators to gauge reception beyond mere numbers.
- **Competitive Analysis:** While not explicitly a feature, the tool can be adapted to analyze competitor channels by simply changing the CHANNEL_ID, offering comparative insights into successful strategies within a specific niche.
- **Trend Identification:** Analyzing topic performance over time (though the current version provides a snapshot) can help spot emerging content trends or shifts in audience interest.

- **Content Audit:** The tool facilitates a systematic review of past content performance, providing data-driven guidance for future production decisions.

Looking ahead, several enhancements could further extend the utility and power of the YouTube Analyzer, indicating its potential for future growth and its ability to adapt to the evolving needs of content creators, ensuring its long-term relevance and value in a dynamic digital landscape :

- **Real-time Analysis:** Integrating capabilities for more real-time or near real-time data fetching and analysis could enable creators to react quickly to trending content or audience shifts.
- **Multi-channel Support:** Expanding the tool to allow simultaneous analysis of multiple channels would facilitate comparative studies and broader market research.
- **Advanced Metrics:** Incorporating additional metrics such as subscriber growth analysis, audience demographics (if supported by the API), or more sophisticated custom engagement metrics could provide even deeper insights.
- **Time-Series Analysis:** Adding functionalities to track performance and sentiment trends over time would allow users to observe the evolution of their channel's performance and audience reception.
- **Interactive Dashboards:** Moving beyond static plots to interactive dashboards could enable users to explore data more dynamically, filter results, and drill down into specific areas of interest.

The YouTube Analyzer, in its current form, already provides a powerful foundation for data-driven content strategy. With potential future enhancements, it stands to become an even more indispensable asset for anyone navigating the complexities of YouTube content creation and marketing.