Week 4: Data Processing and Making Data Model Ready

Yujia Cao, Wendi Yuan, Yuhan Zhao

This week, we are focusing on processing our data based on insights from previous work. We have performed data cleaning, outliers treatment, missing data treatment, data transformation, removing unnecessary columns for modeling, text data preprocessing by removing stop words, and other essential steps. Those actions will help our analysis be more efficient and more accurate.

Our dataset was obtained from 5 different data files and they are categorized by their published regionWe created a new column called *location* for each data file, to identify the region associated with each video. Since those 5 datasets have the same data structure, we are able to merge them due to this feature. We merged the five individual files into one comprehensive dataset, with the same column structure. This step will consolidate our data and provide a unified view, making the analysis more straightforward and insightful.

```
# Add a new column 'location' to each DataFrame
df_ca['location'] = 'China'
df_de['location'] = 'Germany'
df_fr['location'] = 'France'
df_gb['location'] = 'Great Britain'
df_us['location'] = 'USA'

# Concatenate all DataFrames
merged_df = pd.concat([df_ca, df_de, df_fr, df_gb, df_us], ignore_index=True)

# Check the first few rows of the merged DataFrame
print(merged_df.head())
```

Next step is to check if our merged datasets contain any missing values and we need to drop them for future analysis. We have identified 6942 missing values and they all come from *description.* We assume that some videos only contain essential parts for their content but not necessarily have descriptions. After dropping the missing values, we can proceed to the next step.

```
Missing values
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              6942
location                    0
dtype: int64
```

```
Missing values
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              0
location                 0
dtype: int64
```

We use data transformation by creating engagement metrics. In this step, we calculate the weight of each video to help rank all of the videos. Creating engagement metrics is essential in this project because it provides a unified measure of user interaction with the videos, rather than analyzing *likes*, *dislikes* and *comment_count* frequency. This engagement metric combines factors and forms a single variable, allowing a more standardized analysis. This transformation helps capture the overall level of audience engagement and makes it easier to study its impact on other outcomes, such as the number of views. By aggregating different types of interactions into one metric, we reduce the complexity of the analysis and create a more interpretable feature for understanding user behavior.

In this project, the data transformation involves feature engineering, where the predictor variables, such as individual *likes*, *dislikes*, and *comment_count* are transformed into a higher-level feature—'Engagement Metrics'. This transformation is crucial because it enhances the dataset's ability to represent meaningful patterns. For instance, studying the correlation between total engagement and video views becomes much clearer with this unified metric.By providing a single metric that encompasses various user activities, this approach enables us to build more efficient predictive models and conduct statistical analysis. This comprehensive view of user behavior could potentially influence YouTube's recommendation algorithm or video performance.

Then, to clean the data more thoroughly, we firstly scan variables and remove unnecessary variables. These are descriptions of these variables:

video_id: A unique identifier for each video,

trending_time: The specific date when the video became trending,

publish_time: The timestamp of when the video was published,

thumbnail_link: A link to the video's thumbnail image,

comments_disabled, *ratings_disabled*, *video_error_or_removed*: Indicators of whether comments, ratings, or the video itself have been disabled or removed.

```
#drop columns needed
merged_df.drop(columns=['thumbnail_link', 'video_id','comments_disabled','ratings_disabled','video_error_or_removed'

print(merged_df.head())
```

We remove unnecessary columns from a dataset. Columns such as *video_id, trending_time, publish_time, thumbnail_link, comments_disabled, ratings_disabled*, and *video_error_or_removed* may contain metadata that serves little purpose for many types of analysis. For instance, *video_id* is simply a unique identifier for each video and does not contribute to any meaningful insights about the data itself. Each video already gets a unique description and title as identification. Similarly, *thumbnail_link* is a URL to the video's thumbnail image, which is generally irrelevant unless you're performing image-based

analysis or studying visual content. We don't have images to show the visual information so these are not useful.

Another point is that these columns make the data manipulation more complex, removing them will make the data size smaller and easier to analyze. We have a large dataset with almost 200,000 data points after we remove the missing values, and these columns may add up to the processing times of the model since they are also difficult to encode and categorize. Some columns like *comments_disabled, ratings_disabled* and *video_error_or_removed* might be tempting to include, but they often do not provide significant value in determining trends or engagement metrics. We already define the engagement metrics as the combination of *likes, dislikes* and *comment_count*, so there is no need to keep other columns.

After we finish the primary step of cleaning data, we choose to remove stop words and lower case for text preprocessing. Before we process the text data, we look up our dataset to check if there are necessities to transform these text data into simpler forms. There are two major text-type data in our dataset: the *title* of each youtube video, the *description* of youtube video. We notice that there exists some words that are common language and do not contribute little to no value in understanding the overall meaning of texts. They just add noise to but don't contribute any meaning to the analysis we want to do. In other words, since we focus on the youtube trending analysis, we may use methods like sentimental analysis or the text classification in the future to retrieve information, and removing stopwords reduces the dimensionality of data to make it more accessible. The model will only focus on the key words or meaningful words. We also apply the method to change texts into lower cases.Most of the times, the uppercase or lowercase of words don't affect the actual meaning, but if we keep all these words in lowercase, it will ensure the uniformity of texts and avoid word confusion.

```
# Get the list of default English stopwords
stop_words = set(stopwords.words('english'))
stop_words = set(stopwords.words('chinese'))
stop_words = set(stopwords.words('french'))
stop_words = set(stopwords.words('german'))

# Function to remove stopwords and clean text
def clean_text(text):
    # Lowercase the text
    text = text.lower()

    # Remove non-alphabetical characters (retain only letters and spaces)
    text = re.sub(r'[^a-z\s]', '', text)
```

We import the package 'nltk' to remove stopwords and apply lowercases for words. Since the data is from five different countries, the language environment is distinctive. We use four stop_words commands to check in and remove them in English, Chinese, French and german. After we remove the stopwords, we transform the description in lowercases and create new texts for the video description.