# YouTube Recommendation System

Data Mining (2CSDE71)

Team Members :

20BCE058 Devyaniba Chudasama
20BCE104 Jatin Undhad
21BCE531 Pooja Mavadiya

## 1. PROBLEM DEFINITION

YouTube Recommendation System(YRS) is a system wherein we provide a list of categories which consist of genres like Entertainment, News and Politics ,Music, Comedy, Action etc. and based on the category which user selects we suggest the most popular videos in that particular genre and improvise techniques like **collaborative filtering** and use algorithms like **Apriori**, **association rule mining** and **FP-Growth** on the transactional database created based on the watch-history of user and finally display the sequence which says that the user might have watched X video first then Y video and lastly Z video using association mining rules.

## 2. DATASET DESCRIPTION

- The dataset used in YRS is 'Trending YouTube Video Statistics' updated in 2019 from Kaggle.
- The dataset was collected using YouTube API and is a daily record of the top trending YouTube Videos.

- The YouTube New dataset on Kaggle contains information about the trending videos on YouTube for a period of time from November 2017 to June 2018. The dataset includes data from the United States, Canada, Great Britain, Germany, France, Russia, Mexico, South Korea, Japan, and India.

- Since the dataset is very huge we have limited its scope by using only the data collected from the region of 'INDIA'.

- It consists of 16 columns and 37352 rows.

- The dataset consists of two files:

  1) **INvideos.csv**: contains data about the trending videos in India. It includes 16 columns of data such as video title, channel title, publish time, tags, views, likes, dislikes, and comment count, etc.

  2) The dataset also includes a **category_id.json** file that provides a mapping of the category IDs to the category names.

- The headers in the video file are:
  1. video_id (Common id field to both comment and video csv files)
  2. trending_date
  3. title (Displays the title of the video)
  4. channel_title (Displays the channel which published the particular video)
  5. category_id (Can be looked up using the included JSON files, but varies per region so use the appropriate JSON file for the CSV file's country)
  6. publish_time

7. tags (Separated by | character, [none] is displayed if there are no tags)
8. views
9. likes
10. dislikes
11. comment_count
12. thumbnail_link
13. comments_disabled
14. ratings_disabled
15. video_error_or_removed
16. description

- Data type Representation is as follows :

| Categorical | Numerical | Time Series | Boolean |
|---|---|---|---|
| Video_id | Category_id | Trending_date | Comments_disabled |
| title | Views | Publish_time | Ratings_disabled |
| Channel_title | Likes | | Video_error_or_removed |
| Category_name | | | |
| tags | Dislikes | | |
| description | Comment_count | | |

## 3. NOVELTY

In order to implement algorithms like Apriori and FP-Growth we need to have a transactional dataset which consists of user history and the videos they prefer to watch.

Transactional datasets on YouTube videos, such as user purchase behavior, user watch history are not publicly available due to privacy concerns. YouTube does not typically share such data with third parties.

Hence, we created our own transactional data by applying the technique of collaborative filtering. This is a novel work as nobody has implemented Apriori and Fp-Growth based on the collaborative filtered dataset we created.

The user based on the category which he/she selects is provided a list of videos of that particular category. Among which, the user will ask the system to recommend more of such videos.

Based on the cosine similarity, we recommend top 10 videos which are highly correlated to the video user selected. From these top 10 videos obtained, we generate a transactional dataset of 5 users denotes as T1 to T5 and randomly assign the videos they watched out of the 10 recommended videos.

In this way, we generate a user watch history and implement the algorithms of Apriori and FP-Growth.

## 4. ALGORITHM

## Algorithm 1: Data Preprocessing

**Input:**

TD: Transactional Database (collection of transactions, where each transaction is a set of videos watched by user.)

**Output:**

Transform data into format that is more easily and effectively processed in data mining.

**Procedure: YouTube Recommendation System (TD)**

**Step-1**: [Data Cleaning]

MS<- find out missing data from TD

DU<- Removing duplicates from TD

**Step-2**: [handling missing values]

IG variable: ignore the rows

FILL variable: fill the missing data values

IG<-MS

FILL<-MS

**Step-3**: [Data Transformation]

N<- normalizing the data into specified range (-1.0 to 1.0 or 0.0 to 1.0)

AT<- attribute selection (ignoring non-relevant columns like description, publish_time, trending_date , comments_disabled, ratings_disabled etc. )

DT<- Conversion of datatype of features(Converted the datatype of 'category_id' from object to int64 datatype.)

D<- Discretization (replace a row value of numeric attribute by interval levels or conceptual level)

AN<- Addition of a new column 'Category_name'(conversion of numerical values of category_id to categorical values denoting name of category)

**Step-4**: [Data Reduction]

FM<- reduced form of data conversion

**Step-5**: [Data Formatting]

CK<- check the data consistency across features.

**Step-6**: [Analyzing data]

Ensure that the data CK is organized in a way to easy for analysis. (Analyzing the proportion each category holds in the dataset and the videos uploaded by them.)

**Step-7**: [Return Dataset with completion of data preprocessing]

Return transactional dataset TD after performing data preprocessing steps.

**Step-8**: [stop]

End the procedure.

## Algorithm 2: Apriori Algorithm and Association Rule Mining

**Input:**

**TD**: Transaction Database (collection of transactions, where each transaction is a set of videos watched by user)

**Minimum Support Threshold**: minimum number of times a video must appear in the database to be considered frequent.

**Minimum Confidence Threshold**: minimum probability that 'Video-Y' appears in a transaction given that 'Video-X' appears in the transaction, for an association rule 'Video-X' -> 'Video-Y'.

**Output:**

**A list of frequent videos watched together, A list of association rules**

**Procedure: APRIORI (TD)**

**Step-1**: [Computing support for each video]
C1<-support for each video (count number of occurrences)
N<-count total no of videos

**Step-2**: [Deciding on support threshold]
NF<- filter out some videos which are not frequent

**Step-3**: [Selecting and finding support of frequent videos]
For N iterations:
For C1 iterations:
Find out the specific videos which satisfied the minimum support condition.
L1<- store such videos which satisfied the minimum support condition.
End for
End for

**Step-4**: [Repeat for larger set]
Repeat step-1 until all videos will scan at least once through to make a pair of videos and separately stored in C2.

Repeat step-3 until all items will scan at least once through make a pair of videos and separately stored in L2.

**Step-5**: [Generate Association Rule Mining and compute confidence]

k<- 2 power of length(L2)

For k iteration:

Generate association rules from the frequent itemset(i.e. list of videos watched together frequently) by selecting the rules that meet the minimum confidence threshold. For each frequent itemset X, generate all possible non-empty subsets of X, and for each subset Y, generate the association rule X -> Y if the confidence of the rule is at least the minimum confidence threshold.

End for

**Step-6**: [Return list of frequent itemset]

Return the list of frequent itemset i.e. list of videos watched together and the list of association rules.

**Step-7**: [stop]

End Procedure.

**Algorithm 3 : FP-Growth**

**Input:**

**TD**: Transaction Database (collection of transactions, where each transaction is a set of videos watched by user)

**Minimum Support Threshold**: minimum number of times a video must appear in the database to be considered frequent.

**Minimum Confidence Threshold**: minimum probability that 'Video-Y' appears in a transaction given that 'Video-X' appears in the transaction, for an association rule 'Video-X' -> 'Video-Y'.

**Output:**

**A list of frequent videos watched together, A list of association rules**

**Procedure : FP-Growth(TD, minsup**):

**Step 1**: [Counting frequency of each video]

Scan TD once and count the frequency of each video.

Sort the videos in descending order of frequency.

Remove the videos that don't meet the minimum support threshold minsup.

**Step 2**: [Initializing FP-tree]

Create the root of the FP-tree and initialize it to null.

**Step 3:** [Building FP-tree]

For each transaction t in TD:

a. Sort the videos in t in descending order of frequency.

Remove the videos that don't meet the minimum support threshold minsup.

b. Insert t into the FP-tree as follows:

i. Start with the root node.

ii. For each video in t, check if there is a child node with the same video.

If yes, increment the count of that node.

If no, create a new child node for that video and set its count to 1.

iii. Update the header table by linking the new node to the existing node with the same video.

**Step 4:** [Generating Conditional pattern base]

For each video in the header table, do the following:

a. Create a conditional pattern base for the video.

b. Recursively construct the conditional FP-tree for the conditional pattern base.

c. If the conditional FP-tree is not empty, mine it recursively for frequent patterns.

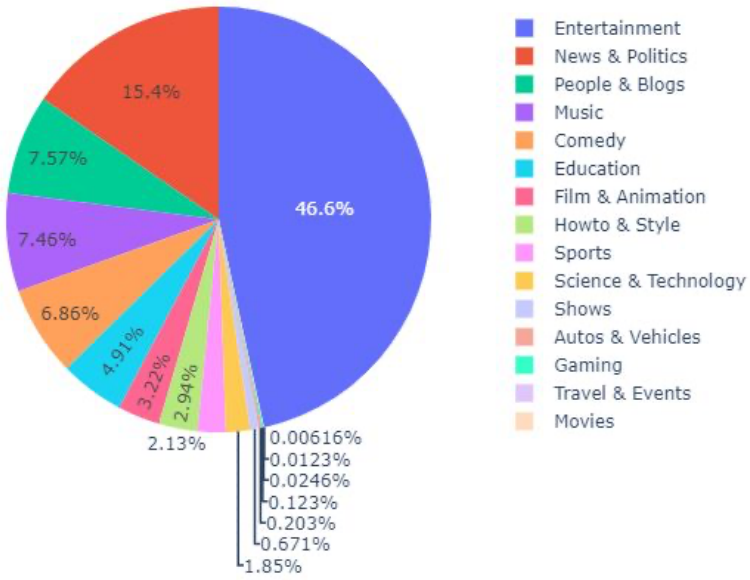
**Step 5**: [Merging & Generating frequent itemsets]

Return the set of frequent patterns found.

**End Procedure**

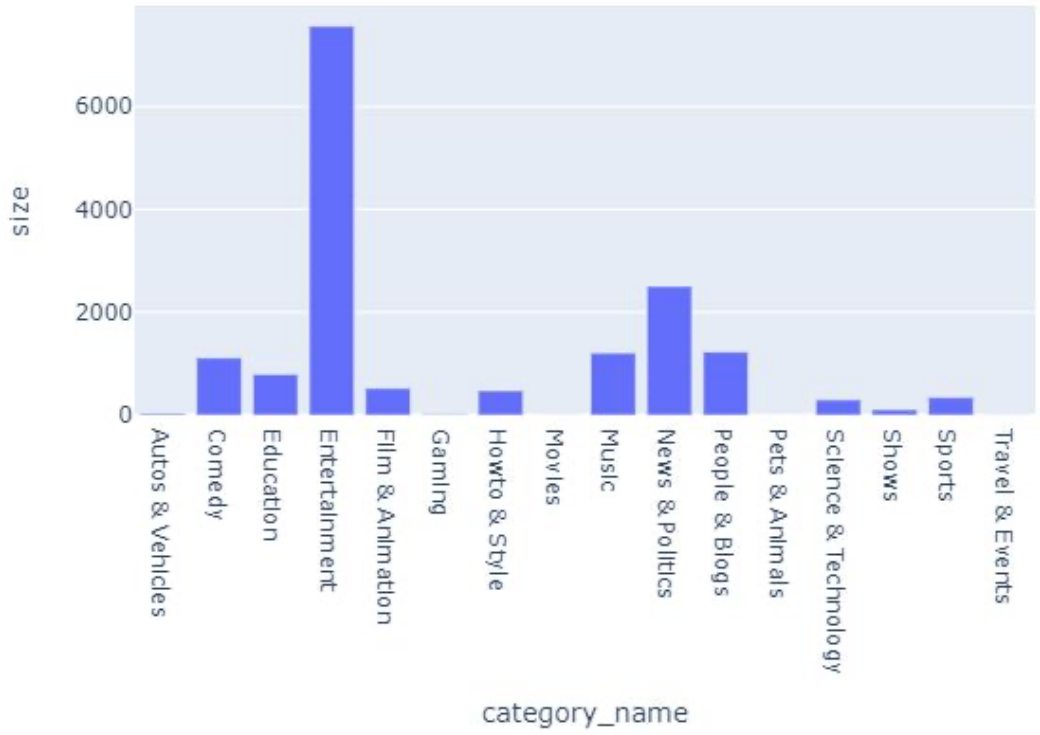
## 5. RESULTS

After pre-processing the dataset we derive following conclusions:



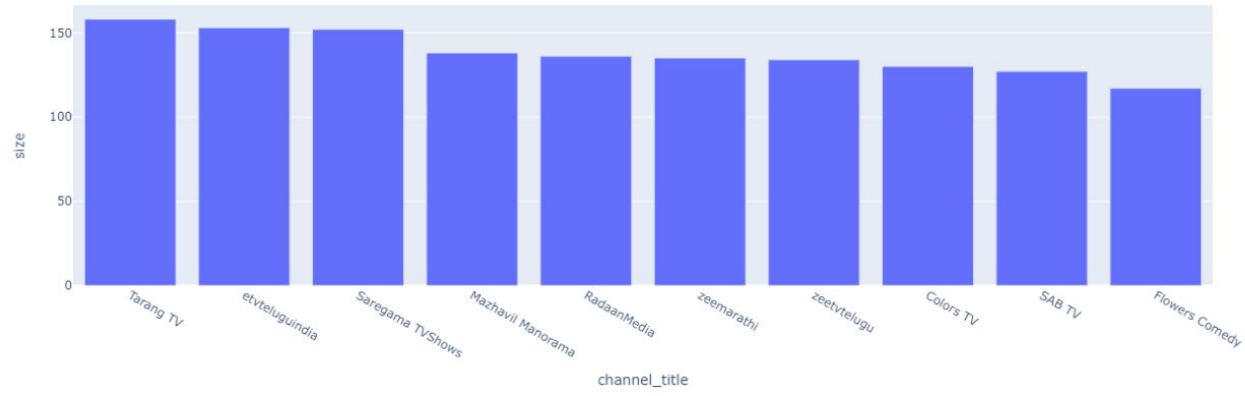

This chart denotes the proportion of each category among the 15 categories provided in dataset. 'Entertainment' category leads the chart by acquiring 46.6% of the proportion while 'Gaming' acquires the least proportion of 0.00616%

## 2. No. Of Videos vs Category




This bar graph denotes the count of videos respective to their categories. Entertainment denotes the highest count of around 7500 videos whereas Gaming, Movies, Pets & Animals, Autos & Vehicles denote the lowest count.

## 3. Top 10 Channels of Entertainment Category.



Since the ‘Entertainment’ Category denotes the highest proportion, we extract the top 10 channels of it where we visualize that ‘Tarang TV’ dominates others in terms of videos they uploaded (around 165) whereas ‘Flowers Comedy’ contains the least count of 120 approximately.

## 4. **Implementation of Apriori and FP-Growth**

Association Rule Mining

Minimum Support count : 70%

Minimum Confidence: 80%

Transaction represented in tabular form

| Transaction | Video 0 | Video 1 | Video 2 | Video 3 | Video 4 | Video 5 | Video 6 | Video 7 | Video 8 | Video 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| T1 | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ |
| T2 | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| T3 | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ |
| T4 | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| T5 | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |

Apriori Method

Frequent Itemsets

| support | itemsets |
|---|---|
| 1.0 | ['Video 0'] |
| 1.0 | ['Video 5'] |
| 0.8 | ['Video 7'] |
| 1.0 | ['Video 0', 'Video 5'] |

FP-Growth Method

Frequent Itemsets

| support | itemsets |
|---|---|
| 1.0 | ['Video 5'] |
| 1.0 | ['Video 0'] |
| 0.8 | ['Video 7'] |
| 1.0 | ['Video 0', 'Video 5'] |

## 5. **Comparison between Apriori and FP-Growth**

Apriori Method

0.008069992065429688 seconds required

FP-Growth Method

0.002994537353515625 seconds required

We observe that FP-Growth algorithm is **faster** than Apriori algorithm as the FP-Growth algorithm requires only two scans

of the dataset, whereas the Apriori algorithm requires multiple scans. This makes the FP-Growth algorithm faster and more efficient than the Apriori algorithm, especially for large datasets with high dimensionality.