Trending Youtube Video Analysis

Team Member: Ashutosh Rawat, David Ma, Mitali Bante, Tracy Cui, Wei-Lun Tsai

Background and Context:

As Youtube becomes more and more popular nowadays, our group is interested in exploring the top trending videos and learning more about various characteristics of popular videos. We found this dataset on Kaggle that is connected to Youtube API and contains a list of data on daily trending YouTube videos. This dataset includes data for several months and for different regions/countries. Also, it contains basic and relatable information about each video (for example, video title, tags, publish time, tags, views, likes and dislikes, etc.).

Dataset:

This dataset includes several months of data on daily trending YouTube videos. Data is included for 10 different countries with up to 200 listed trending videos per day. Each region's data is in a separate file. Data includes the video title, channel title, publish time, tags, views, likes and dislikes, description, and comment count.

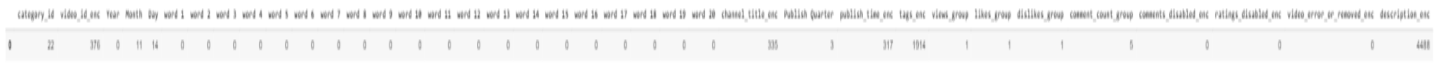
USA Category identification:

15 Features were preprocessed for this prediction. Label encoding was mostly used because there was no distinct hierarchy for the given feature data and one hot encoding would have given a massive dataset which is not computationally efficient.

Feature preprocessing methods:

- *Comments disabled*, *Ratings disabled*, *Video Error or Removed*: Binary Label Encoding (1 or 0):
- *Video ID, Tags, Channel title, Description*: Multiclass Label Encoding (>2 categories)
- *Views, Likes, Dislikes, Comments*: Sector Assigning (5 different sectors were assigned based on count)
- *Thumbnail Link*: Dropped
- *Trending date*: Split feature into Year, Month, and Day and add these as columns
- *Title*: 20 context words were searched from the top 50 words given title and if exists, a label of 1 was assigned for each sample
- *Publish time*: Column of published quarters (early morning, morning, afternoon, evening) were assigned and label encoding of the exact published time was performed

<u>Final Dataframe of data</u>



	category_id	video_id_enc	Year	Month	Day	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15	word 16	word 17	word 18	word 19	word 20	channel_title_enc	Publish Quarter	publish_time_enc	tags_enc	views_group	likes_group	dislikes_group	comment_count_group	comments_disabled_enc	ratings_disabled_enc	video_error_or_removed_enc	description_enc
0	22	376	0	11	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	335	3	317	1914	1	1	1	5	0	0	0	4488

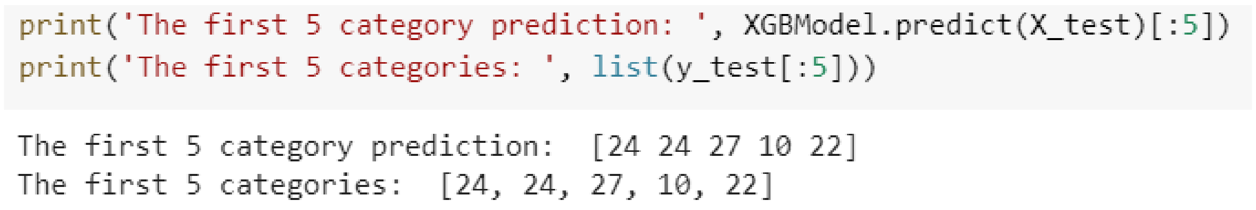
As this is a multilabel classification problem, trees and boosted trees were used to evaluate the performance. A data split of 8 (Development): 2 (Test) was used. Hyperparameter tuning was conducted with development set by adjusting the following parameters for each model: Random Forest (max_depth, max_leaf_nodes), Decision Trees (max_depth), Gradient Boosting classifier (n_estimators, max_depth), Histogram Gradient Boosting classifier (max_leaf_nodes, max_depth), XGBoost classifier (n_estimators, max_depth). Due to the large computational time and testing, only the evaluation of the optimal parameters are shown in the code file.

<u>Model Performance Table</u>

Models	Test Set Classification Accuracy
Random Forest	0.961
Decision Trees	0.983

Gradient Boosting Classifier	0.986
Histogram Boosting Classifier	0.987
XGBoost Classifier	0.990

The predicted results of the first 5 videos using our best model:



```
print('The first 5 category prediction: ', XGBModel.predict(X_test)[:5])
print('The first 5 categories: ', list(y_test[:5]))

The first 5 category prediction:  [24 24 27 10 22]
The first 5 categories:  [24, 24, 27, 10, 22]
```

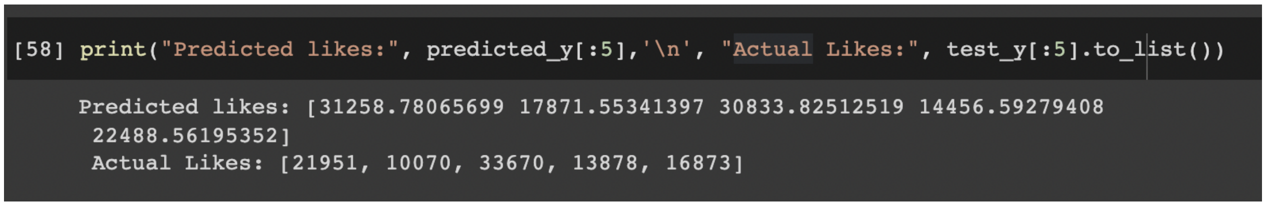
Regression for number of likes/comments:

For this section, we focus on predicting the number of likes and comments for a given video. First for data cleaning, instead of looking at the publish time as a whole, we divided it into year, month, and the day number of the year (out of 365 days), and hour of the day. We also calculated the difference between the publish day and the trending day. In this way, we will be able to capture the timeframe when the video is published and use this as part of our prediction features. On top of that, we also used categories, number of views, number of dislikes, comments_disabled, ratins_dsiabled, video_error_or_removed as features for our model.

We used 4 different linear models (simple linear regression, ridge regression, lasso regression, and elastic-net regression), and we tuned hyperparameters for each of them(except simple linear regression) in order to get better results for each model. However, as shown in the table below, for both the number of likes and comments, all 4 models give similar accuracy. We chose to use the simplest one (Ridge regression model) to predict because it's more efficient.

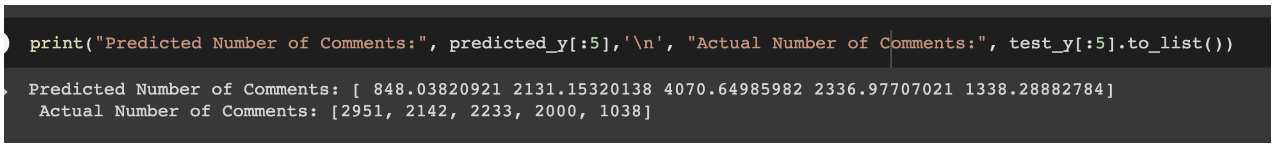
Model	Likes Accuracy	Comments Accuracy
Simple Linear Regression	0.8844	0.8224
Ridge Regression	0.8805	0.8316
Lasso Regression	0.8805	0.8314
Elastic-net Regression	0.8805	0.8316

The predicted results for the first 5 videos in the test set are as follows:



```
[58] print("Predicted likes:", predicted_y[:5],'\n', "Actual Likes:", test_y[:5].to_list())

     Predicted likes: [31258.78065699 17871.55341397 30833.82512519 14456.59279408
      22488.56195352]
      Actual Likes: [21951, 10070, 33670, 13878, 16873]
```



```
print("Predicted Number of Comments:", predicted_y[:5],'\n', "Actual Number of Comments:", test_y[:5].to_list())

Predicted Number of Comments: [ 848.03820921 2131.15320138 4070.64985982 2336.97707021 1338.28882784]
 Actual Number of Comments: [2951, 2142, 2233, 2000, 1038]
```

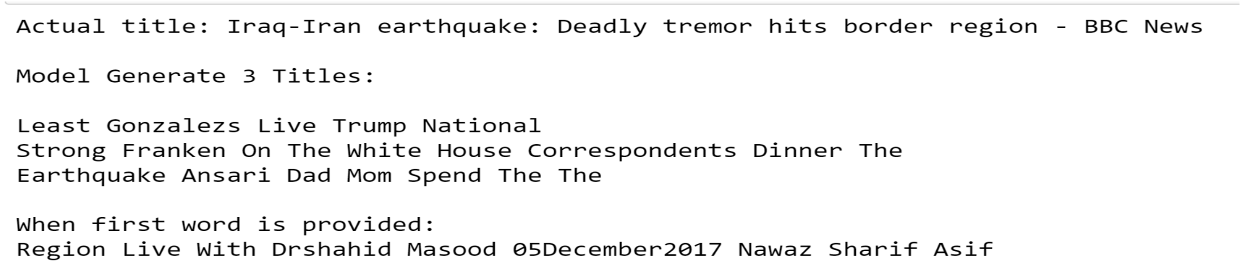
Title Generation:

Introduction: In order to generate titles for youtube videos, we make use of Natural Language Processing (NLP) and Deep Learning techniques. To learn the hidden sequential relations of the words we need more data. So, we made use of data not only from the United States (US) but also from Canada (CA) and Great Britain(GB) as these countries too have English titles.

Data Cleaning and processing:

- Data from US, CA, and GB was concatenated
- Since one single video can be trending for multiple days, the duplicates based on video_id were removed
- Separate data based on video category to train on category-specific data
- For each model (based on category), generate n-grams for the titles. Eg: If the title is- ‘Tonight Big Show’. The n-grams are: [Tonight, Tonight Big, Tonight Big Show]

Title generation using LSTM model: As LSTM models can be used to generate sequential data and have long-term memory compared to the generic RNN model, we utilized the LSTM layer. The architecture consists of the input layer (to take in the sequence of words), LSTM layer (to learn sequential word occurrence), dropout (to avoid overfitting), output layer (to predict the next word.
Output: After the model is trained based on n-grams in the title, we go through the description of the video, remove special symbols, characters, and URL links from the text. Get the top 3 high-frequency words and call the generator model to predict the next words for the title.



```
Actual title: Iraq-Iran earthquake: Deadly tremor hits border region - BBC News

Model Generate 3 Titles:

Least Gonzalezs Live Trump National
Strong Franken On The White House Correspondents Dinner The
Earthquake Ansari Dad Mom Spend The The

When first word is provided:
Region Live With Drshahid Masood 05December2017 Nawaz Sharif Asif
```

Fig: Title Generation for a random video from News and Politics Category

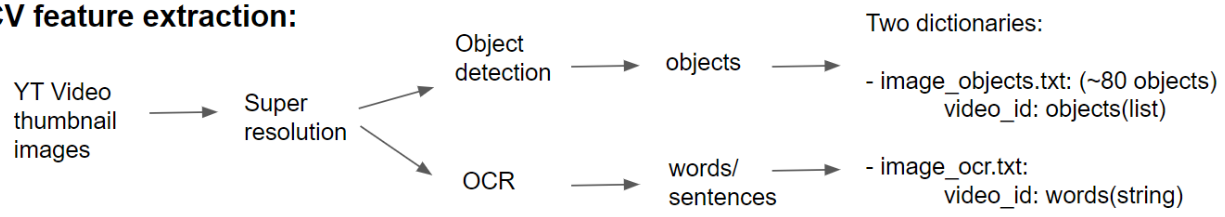
Thumbnail content recommendation:

Overview: This section focuses on building a thumbnail content recommendation model for video producers to determine which objects they could put in their thumbnail to get a higher chance of making the video a trending one. Given video features (title, tags, descriptions, and category), the model recommends suitable objects, such as person, car, tie, chair, and/or intro sentences. The result shows that the system can predict six labels with **90.29% accuracy (f-1 score: 64.66%)**.

Text and image - feature extraction and data preprocessing: For image features, we use an RDN (Residual Dense Network) to do image super-resolution, followed by a pre-trained resnet50 for object detection. Also, we use Tesseract - an OCR (optical character recognition) engine - to extract words/sentences. For text preprocessing, we do stop word removal, lemmatization, proper noun generalization, punctuation and case unification, tokenization, etc.

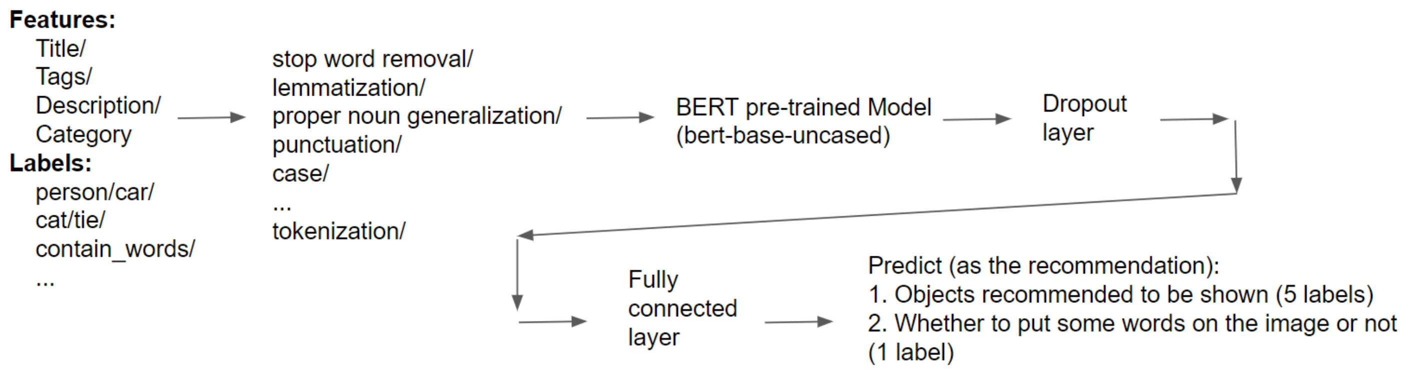
Multi-labeled classification using BERT: This task is intrinsically a multi-labeled text classification problem with its labels being objects in images. Our architecture contains a pre-trained BERT, a dropout layer, and a fully connected layer. It is trained with 50 epochs and uses a three-way hold-out for model selection.

1. CV feature extraction:



YT Video thumbnail images
Super resolution
Object detection
objects
OCR
words/ sentences
Two dictionaries:
- image_objects.txt: (~80 objects) video_id: objects(list)
- image_ocr.txt: video_id: words(string)

2. NLP + BERT for multi-labeled classification:



Features:
Title/
Tags/
Description/
Category
Labels:
person/car/
cat/tie/
contain_words/
...
stop word removal/
lemmatization/
proper noun generalization/
punctuation/
case/
...
tokenization/
BERT pre-trained Model (bert-base-uncased)
Dropout layer
Fully connected layer
Predict (as the recommendation):
1. Objects recommended to be shown (5 labels)
2. Whether to put some words on the image or not (1 label)

References:
https://www.mygreatlearning.com/blog/label-encoding-in-python/
https://thatascience.com/learn-machine-learning/label-encoding/
https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/
https://thecleverprogrammer.com/2020/10/05/title-generator-with-machine-learning/