Data Analytics Pipeline

Project: Youtube Channel Recommendation System

- Aashna Kanuga (adk2159)
- Akshata Patel (amp2313)
- Kiran Saini (ks3630)
- Neha Saraf (ns3308)

1. **JIRA link:**
   https://toydemoproject.atlassian.net/jira/software/projects/YCR/boards/23

2. **GitHub link:**
   https://github.com/nehasaraf1994/DataPipeLineProject

3. **MVP Description :**

A. Description of the product

Our product recommends Youtube channels to the user based on user inputs such as product category and video category. The top 5 channels are recommended on the basis of their similarity to the product and their popularity. The channel recommendations can be used by the user to promote and advertise their product.

B. Individual APIs

a. Training Service:

We used the pre-trained word embeddings from the Google News word2vec model for our MVP. We implemented this using the gensim package in Python. The trained model is saved in a pickle file to be used by the prediction and recommendation API. The training function is encapsulated in a Flask API that will be called over certain periods of time to re-train the model on the dynamically updated data. (For the MVP as the data is static, it only needs to be called once)

- ➔ train_model() method: loads the pre-trained Google news word2vec model and returns it.
- ➔ save_model() method: calls the train_model() function and saves the returned word2vec model into a pickle file. App routing is used to bind the “/training_service” URL to this method. As a result, if a user visits http://localhost:5000/training_service URL, the output of the save_model() function will be rendered in the browser i.e. the

model will be saved in the local folder and the text would be returned.

b. Prediction and Recommendation Service

This service is used to read in all the data (YouTube information, User choices as well as the stored model - saved using the training service above), and predict the most similar videos to the user's product and video category. Then using these predictions, we group by channel and pick the top 10 most similar channels. From these top 10, we pick the top 5 channels that have the highest difference in their average likes and dislikes and send this data to the dashboard service.

- ➔ LoadModel(): defines get_stored_model() which loads the pickled file of the pre-trained word2vec model and returns it

- ➔ GetPredData(): defines the get_data() method, which loads data from the mysql database returns it.

- ➔ PredictHandler(): This class is responsible for getting the similarities of the user data against all videos. The following methods are defined:
  - ◆ get_text(): This method reads gets the youtube data from the get_data method of the GetPredData class, and returns it

  - ◆ get_similarity_score(): This method read in the product category, video category from the user data and category name and tags of each youtube video, and calculates the similarity between them. We get similarity values for youtube category name with the user's product and video category. Then, we get the similarity of each tag with the user's product and video category, from which we take the top 5 most similar tags and get the mean similarity for both. Thus, we have 4 similarity values, from which we get 2:
    - User product category with youtube category name and tags (then we take its average)
    - User video category with youtube category name and tags (then we take its average).

    Now we take a weighted average of these two values to get the final similarity value for one video (for the MVP both

product and video category have equal weights, in the next iteration the user can provide them) and return it.

- predict(): This method takes the user input from the dashboard (product and video category) as parameters, calls get_text() to get the youtube data, and then calls the get_similarity_score() method to find the similarity score with these user inputs.

➔ RecommendationHandler() : This class is responsible for recommending appropriate channels to the user depending on its product and video category inputs. It has the following method:

- recommend_channel(): This method calls the predict() method. The similarity scores returned by the predict method are grouped and averaged by the channel title. The top 10 of these channel titles are taken. These channel titles are sorted on the basis of the difference in their total number of likes and dislikes. The top 5 channel titles from this sorted list are returned from this method.

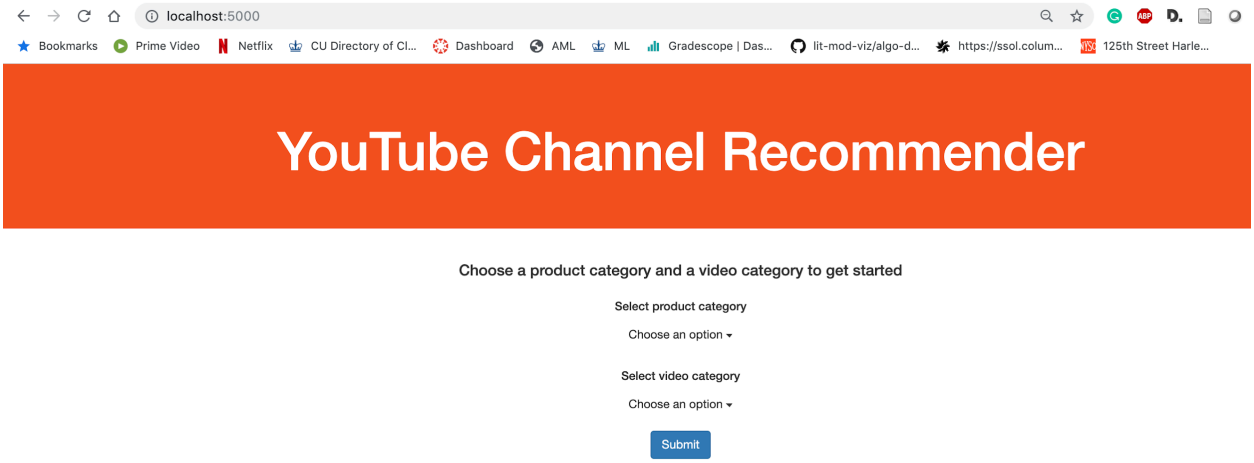
C. Creating the Dashboard :

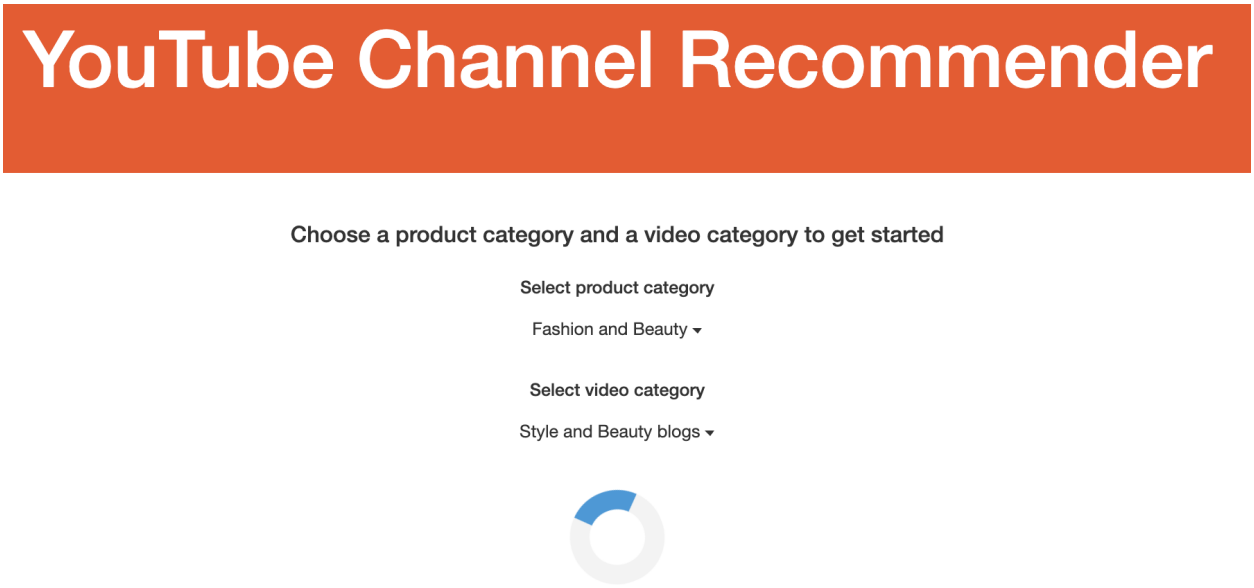
➔ main.py : Created a Flask App which renders an HTML template i.e our dashboard.

➔ get_product_data.py : It contained code to create a GET API using Flask with two input parameters: product and video category and outputs top 5 recommendations by calling the RecommendationHandler().

➔ home.html : We have created a form to take 2 inputs from the user : product category and video category. When the user clicks on the “Submit” button, the API of the Get_product_data python file is called which returns the top 5 recommendations and they are displayed on the dashboard.

Screenshots :





YouTube Channel Recommender

Choose a product category and a video category to get started

Select product category

Fashion and Beauty

Select video category

Style and Beauty blogs

Submit

YouTube Channel Recommendations

The top 5 recommendations are:

No.	Channel
1	Safiya Nygaard
2	Kylie Cosmetics
3	Glam&Gore
4	Boldly
5	ConnorFranta