Tejaswi Rao
trao@stevens.edu

Inside.com - ML/AI Engineer Proficiency Evaluation Exercise

Project: Create a specification document for AI based news curation from trending YouTube business content.

Methodology:

1. Data Collection:

1) Detail how you would gather a dataset of trending business news videos on YouTube.

Ans- There are two ways to gather the data, the long way is to web scrape the data from third party websites using Selenium or Beautiful Soup. But this can be of different perspectives to that of the original ones. So instead of that, I am going to be using the YouTube API to gather the dataset. To access the YouTube API, one needs to have a project running on the GCP.
-You can search for YouTube api and enable it inside your project from the search bar

-Create credentials and create API key

-The generated API key can be used to gather the YouTube data (its totally free)

-Here is my API key I used to build this project **api_key = 'AIzaSyD0uAJTI0jlxQ-tB_buCNxEnhtCTceFnyA'**

2) Describe how you would ensure the dataset includes a variety of business-related topics and covers current trends.

Ans- The first thing that I would do is to declare the keywords or queries

```
# Define a list of business-related keywords
business_keywords = ['business', 'finance', 'economics', 'stock market', 'startups', 'breaking news', 'current affairs',
```

When fetching videos from the YouTube API, you can specify the relevance parameter to filter the search results.

I can also fetch only the videos that are trending and process them for news. This is an added feature of YouTube API that is super helpful, but for my code, I just considered it based on the default relevancy score.

Also, There is an option of choosing the category of videos you need to consider for your study, In our case, since we are mostly interested in current affairs and recent trends, we can switch on the you tube video category ID so that we are only dealing with News category, The category ID for News is 25, The reference for this is at https://mixedanalytics.com/blog/list-of-youtube-video-category-ids/

So this way, we focus mostly on news and within that category, we will be looking for Business related news.

3) Cover what methodology you would use to capture the relevant information from the chosen dataset and what information your dataset would consist of. Explain the relevance of your criteria determination.

Ans- You Tube API allows us to capture mainly the title and description and other metadata from the video like Video Id, Title, Upload date, View count, description, comments, impressions.

Once I have these data, I would use NLP to capture a couple of additional things that I can use later

I will capture the sentiment score and the recency score. The importance of this will be touched upon later.

2. Preprocessing and Feature Extraction:

1) Explain how you would clean and preprocess the collected data.

Ans- The necessary libraries are imported, including **re** for regular expressions, **nltk** for natural language processing tasks,and **WordNetLemmatizer** for word lemmatization. The WordNet resource from NLTK is downloaded using the nltk.download('wordnet') command. WordNet is a lexical database that provides access to lemmas and semantic relationships between words.

For preprocessing, I defined a function which involves multiple steps, While dealing with Text data, its really important to normalize the data.

- Initially I removed all special characters, punctualtions and symbols
- Convert all the text into the lowercase
- Break down the text into individual words, it makes it easier to process and analyze
- Remove the stop words (‘the’, ‘is’, ‘and’), it doesn’t define the soul of the statement, hence its better to remove them
- As a last step, I used Lemmatization technique, which reduces the words to their base form (eg – reading becomes read), it helps reduce vocab size
- Final step is to join the shortened words together to form a sentence

```
def preprocess_text(text):
    # Remove special characters, punctuation, and symbols
    text = re.sub(r'[^\w\s]', '', text)

    # Convert text to lowercase
    text = text.lower()

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [token for token in tokens if token not in stop_words]

    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]

    # Join the tokens back into a string
    preprocessed_text = ' '.join(lemmatized_tokens)
```

2) Describe what relevant features you would extract from the dataset's metadata and why.

Ans- From the Metadata, I extracted Title_length, description length, word_count, keyword counts (business, startups, finance etc)

Note – Apart from these, I can also extract the Audio Transcripts("captions.list") and video captions ("contentDetails.caption")(user uploaded or auto generated), But unfortunately, Since I am using the Free API, It is not allowing me to access those, to bypass this, I need to use **OAuth2 authentication to authenticate any requests.** OAuth2 authentication involves obtaining access tokens that provide authorization to access protected resources. (Normal API doesn't work)

The main reason behind extracting these features is to approach this problem as a structured data problem (with labels)

The main reason behind calculating keyword counts is to label them

```
Title: Session 01: Objective 1 - What Is Corporate Finance?
News Content: The Finance Coach: Introduction to Corporate Finance with Greg Pierce Textbook: Fundamentals of Corporate F
inance Ross, ...
Features: {'title_length': 52, 'description_length': 123, 'word_count': 31, 'keyword_business': 0, 'keyword_finance': 1,
'sentiment_score': 0.0}
```

The extracted features look like this, here we can see, it has keyword finance = 1, so this will be labelled as a finance video, in case there is a tie between the counts, it will be labelled as both. This result is mainly by counting the keywords, the alternate approach for the same is TF-IDF,
TF-IDF (Term Frequency-Inverse Document Frequency) Calculates the importance of words in the snippets and titles based on their frequency in the specific video and across the entire dataset. TF-IDF gives more weight to rare words that are more informative.

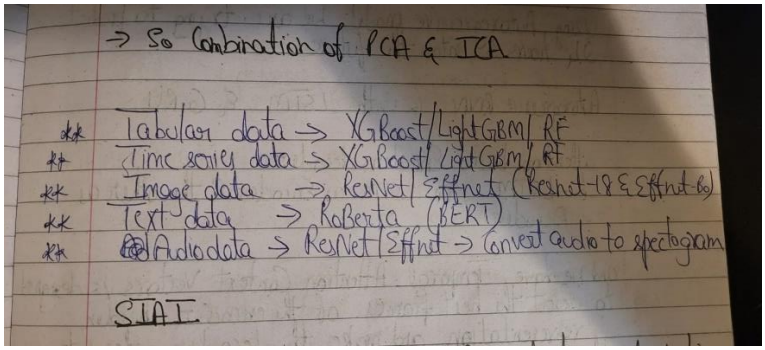
Another reason to extract these features is to calculate the sentiment score of the video. The sentiment score indicates whether the content is positive, negative, or neutral. If the value is close to -1, then its termed as negative, if its close to 0, then its termed as Neutral, if its close to 1, then its termed as positive. This will play an important role in deciding the nature of the video or recommending the video at the later stages.

3. Machine Learning Model Selection:

1) Detail how you would research and select the most appropriate ML algorithms for content curation.

Ans- Since I have been working in the Research field, my first step would be to refer to research papers who were working closely on the field that I am trying to come up with a solution to, would give me an idea on what models to consider and what not to. Ideally, at any time, I will be working on a couple of algorithms (ideally starting from simple statistical models all the way to neural networks), I personally believe in fine tuning and hence I am more inclined towards trial-and-error technique of testing models. Apart from that, it also depends upon the data that I am using, I have a few go to algorithms which works good with certain type of data, so I will be testing out those first.

Coincidentally, I have a note which highlights which algorithms perform better on the various types of data. So, I would be testing out these algorithms first.



2) Explain what algorithms you would consider for classification, clustering, recommendations, etc. — detail specific requirements and constraints.

Ans- There are two ways again to look at this, based on my conversation with Asheesh, He said that users would be subscribing to the specific classes, and we do not have to automate anything. If this is the case, Then We first have to classify the news based on the Classes (business, finance, startups etc)

Once we classify it, we need to cluster it based on the classes.

And recommend these cluster based on the user choices. This is the traditional way and I have used XGBoost for classification, K means for clustering and Content based filtering for Recommendation. The code for this approach is available in the Jupyter notebook.

The alternate way to do it is to use TFIDF vectorizer or matrix factorization model

```
classifier = XGBClassifier()
classifier.fit(X, y)

kmeans = KMeans(n_clusters=5)
kmeans.fit(X_train)

#option 1
svd = TruncatedSVD(n_components=10)
svd.fit(X_train)

#option 2
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)
```

You can further modify this code to incorporate BERT for clustering or recommendation tasks by leveraging BERT embeddings or similarity measures. BERT embeddings can be used to represent video titles or descriptions, and you can apply clustering algorithms or similarity measures to group similar videos or recommend relevant videos based on user choices.

4. Training and Evaluation:

1) Explain how you would split the dataset into training and testing sets.

Ans- Randomly split it into 70-30 or 80-20 using train_test_split function

3) Describe how you would train the selected ML model using the training set

Ans-
-Preprocess the data by Applying any necessary preprocessing steps to your training set. This may include cleaning the data, handling missing values, normalizing features, or encoding categorical variables.

-Define the model and define the hyperparameters, for example if I am using K means clustering, you need to define the clusters `kmeans = KMeans(n_clusters=5)` , similarly if I am using Deep Learning, I will have to define, regularization parameter, learning rate, epochs, different layers, dimensions of the layers etc.

-In python, its fairly simple, you just Fit the model to the training data using the fit method.

-Evaluate the model: After training, assess the model's performance on the validation set or the test set

-Iterate and tune: Based on the evaluation results, I may need to iterate on steps 2-4 by adjusting hyperparameters, trying different algorithms, or applying feature engineering techniques to improve the model's performance. This process is known as hyperparameter tuning.

3) Detail how you would evaluate the performance of the training model & describe what metrics you would use.

Ans- For Classification and especially for an imbalanced data, Accuracy wouldn't do justice.(Eg- 90 abnormal case and 10 normal case, even if the model predicts all the data as abnormal, the accuracy of the model is 90% which is not the case), To avoid this, I would like to try Precision, Recall and F1 score. In simple words Precision, recall, and F1 score provide a more nuanced evaluation of model performance, especially in scenarios where class imbalance or the relative costs of different types of errors are a concern.

Now, coming back to our problem, since it is multi class classification, we can plot a confusion matrix to visualize the model's predictions. We can not only visualize, but also get the model performance for each classes along with its performance metric

As far as the clustering is considered, I would analyse the clustering results based on intrinsic evaluation measures like silhouette score or within-cluster sum of squares.

Additionally, you can visually inspect the clustering results using scatter plots or other visualization techniques.

To evaluate the Recommendation system, we can use a few techniques, some of them are,
Precision and Recall: Precision and recall are widely used metrics for evaluating recommendation systems. Precision measures the proportion of recommended items that are relevant to the user, while recall measures the proportion of relevant items that are recommended.

Mean Average Precision (MAP): MAP is another popular metric for evaluating recommendation systems. It considers the ranking of the recommended items and calculates the average precision at different positions in the list.

Normalized Discounted Cumulative Gain (NDCG): NDCG is a metric that measures the ranking quality of the recommended items. It considers the relevance of the items and their positions in the recommendation list.

A/B Testing is also a good method to measure the effectiveness of the model

5. Content Filtering and Ranking:

1) Explain how you would develop a system to filter and rank the trending business news videos based on their relevance and quality.

Ans- To take into account the relevance and quality apart from the inbuilt relevant factor from the API, I have written a function, which takes into account the the keywords, recency and engagement. Each of these have a set of weightage to begin with (Keyword weightage is 50%, recency weightage is 30% and engagement weightage is 20%)

```
# Get video information
title = video['snippet']['title'].lower()
description = video['snippet']['description'].lower()
published_at = video['snippet']['publishedAt']

# Calculate keyword relevance score
keyword_relevance = 0
business_keywords = ['business', 'finance', 'economics', 'stock market', 'startups']
for keyword in business_keywords:
    if keyword in title or keyword in description:
        keyword_relevance += 1

# Calculate recency relevance score
recency = calculate_recency(published_at)
recency_relevance = 1 / (recency + 1)  # The higher the recency, the lower the relevance score

# Calculate engagement relevance score (dummy calculation)
engagement_relevance = video['statistics']['viewCount'] / 1000

# Calculate overall relevancy score
relevancy_score = (
    keyword_weight * keyword_relevance +
    recency_weight * recency_relevance +
    engagement_weight * engagement_relevance
)

return relevancy_score
```

Data Collection:

-Gather a dataset of trending business news videos from a reliable source, such as YouTube API, as you mentioned.
-Ensure that the dataset includes a diverse range of business-related topics and covers current trends. This can be achieved by querying relevant keywords, channels, or categories.

Data Preprocessing:

-Clean and preprocess the collected data by removing special characters, punctuation, and symbols.
-Convert the text to lowercase, tokenize it into words, and remove stopwords to focus on important keywords.
-Lemmatize the tokens to reduce them to their base form for better analysis.
-Additionally, calculate sentiment scores using techniques like TextBlob to capture the sentiment of the video content.

Feature Extraction:

-Extract relevant features from the video metadata and text data. Some of the features you can consider are:
Title length: Length of the video title.
Description length: Length of the video description.
Word count: Total number of words in the title and description.
Keyword relevance: Presence of business-related keywords in the title or description.
Sentiment score: The sentiment polarity of the video content.
Recency: Time elapsed since the video was published.
Engagement metrics: Likes, comments, view count, or other metrics indicating user engagement.
-These features will help in assessing the relevance and quality of the videos.

Relevancy Score Calculation:

-Calculate a relevancy score for each video based on the extracted features and their respective weights.
-The relevancy score can be calculated by combining the scores from different factors (e.g., keyword relevance, recency relevance, and engagement relevance) using appropriate weights.
-The relevancy score helps in ranking the videos based on their relevance to business news.

Quality Assessment:

-To evaluate the quality of the videos, you can consider additional factors such as production value, credibility of the source, presenter's expertise, and user feedback.
-Incorporate quality assessment metrics into your system to filter out low-quality or unreliable videos.

Filtering and Ranking:

-Sort the videos based on their relevancy scores in descending order to prioritize the most relevant ones.
-Apply filters based on user preferences, such as specific business topics, duration, or language.
-Present the filtered and ranked videos to the users in a user-friendly interface.
Continuous Learning and Feedback:

-Incorporate feedback mechanisms where users can rate or provide feedback on the curated videos.
-Utilize user feedback and engagement data to continuously improve the relevancy and quality of the recommendations.
-Periodically update the ML models and algorithms to adapt to changing trends and user preferences.
By developing such a system, we can effectively filter and rank trending business news videos based on their relevance and quality, providing users with valuable and personalized content. The relevancy score, as explained in the previous answer, serves as a crucial metric to assess, and rank the videos, considering factors like keyword relevance, recency, and engagement. The calculated relevancy score helps prioritize videos that align with user interests and trending business topics, enhancing the overall curation process.

2) Explain how you would use the trained ML model to predict the relevance of new videos and rank them accordingly.

Ans-

Data Preprocessing:

Apply the same preprocessing steps to the new video data as done during training and feature extraction. This includes cleaning the text, converting it to lowercase, tokenizing, removing stopwords, and lemmatizing the tokens.

Feature Extraction:

Extract the relevant features from the new video metadata and text data, following the same feature extraction process used during training. Ensure that the extracted features are consistent with the ones used to train the model.

Feature Encoding:

Encode the extracted features into a format compatible with the ML model. This might involve scaling the features or transforming categorical variables into numerical representations.

Predicting Relevance:

Pass the preprocessed and encoded features of the new video through the trained ML model.
The model will predict the relevance score or the class label of the new video based on the learned patterns and relationships from the training data.
If you have used a regression model, the predicted relevance score will indicate the level of relevance. If you have used a classification model, the predicted class label will determine the relevance category (e.g., business, finance, startups).

Ranking:

Once you have obtained the predicted relevance scores or class labels for the new videos, sort them in descending order based on their relevance.
Assign a rank or score to each video, indicating its position in the ranking list. This can be based on the predicted relevance score or the order of the videos in the sorted list.

Post-processing and Presentation:

Apply any post-processing steps or filters to refine the list of ranked videos based on additional criteria or user preferences.
Present the ranked videos to the users in a user-friendly format, such as a curated list or a recommendation system.

It's important to note that the effectiveness of the ML model's predictions and ranking depends on the quality and representativeness of the training data. Continuous monitoring and periodic retraining of the model with new data will help improve its performance over time. Additionally, incorporating user feedback and engagement metrics into the model can enhance its ability to predict relevance and provide more personalized recommendations.

Apart from this, If we decide to productinalize this model, then handling real time data would need a slightly different approach, That is where Spark comes into picture. I have included a sample pseudocode on how to use this in the jupyter notebook, but in general, Pyspark can be used in

-Data Processing and Transformation: PySpark provides a distributed computing framework for processing and transforming large-scale datasets efficiently. It can handle the processing of video metadata and text data in parallel, enabling faster data preprocessing, feature extraction, and encoding.

-Scalability: PySpark's distributed computing capabilities allow you to scale your system to handle large volumes of data. It can leverage a cluster of machines to process data in parallel, making it suitable for handling real-time streaming data or large batch processing tasks.

-Real-Time Processing: PySpark can integrate with streaming frameworks like Apache Kafka or Apache Flink to handle real-time data ingestion and processing. This allows you to continuously process and update the trending videos based on the latest data.

-Performance Optimization: PySpark optimizes data processing tasks through its underlying distributed computing engine. It leverages in-memory computation and lazy evaluation to optimize data transformations and minimize unnecessary data shuffling, resulting in faster processing times.