YouTube Comment Analyzer:

Analyzing Comments for Genre Classification

BAX 452 Machine Learning Final Project

Group 7

Kumar Kishalaya

Yuna Kim

Omkar V. Shirigannavar

Executive Summary

Youtube has a sophisticated recommender system to find the most similar videos but they don't use the *textual data from their comments section* in their recommendation algorithm. The idea here is to project Youtube videos from different genres onto a vector space using their comment section data using various methods of **word embedding** and try to see if similar videos are clustered together based on their comments.

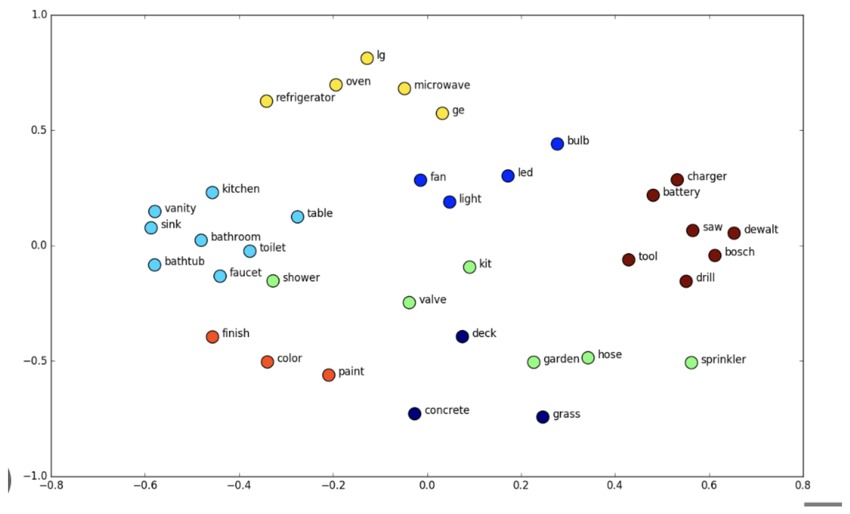
Since we are going to be performing NLP tasks, we will also be exploring the world of word embeddings. We will discuss in detail about two specific embedding algorithms **- word2vec and doc2vec** and how each performs in this specific case. We aim to use the best performing embedding technique for our use case to train classifier algorithms to **predict the genre** of a video **given their youtube comments**.

Background, Context, and Domain Knowledge

Before we move ahead let's get some ideas about **word embedding**. Word embedding represents words as vectors in a high-dimensional space, making it easier to perform mathematical operations and perform similarity analysis.

What is Word2Vec?

Word2Vec is a specific type of word embedding designed to represent words in a high-dimensional space such that similar words are closer together. The effectiveness of Word2Vec comes from its ability to group together vectors of similar words. The below projection of words representing household items will make it a bit more clear about how output from word2vec embeddings display this relation.



What is Doc2Vec?

Doc2Vec, also known as **Paragraph Vector,** is an extension of Word2Vec that learns vector representations of documents rather than words. Doc2Vec was introduced in 2014 by a team of researchers led by Tomas Mikolov. Doc2Vec learns vector representations of documents by combining the word vectors with a document-level vector. The primary idea behind Doc2Vec is to associate a unique vector representation with each document in a corpus. This vector is learned during the model training process, similar to how Word2Vec learns word embeddings.

What should we be using in our problem?

Our problem statement is to associate a youtube video with a unique n-dimensional vector using the corpus of all comments on that video (consider the corpus of comments as the document). Since word2Vec gives embedding for each word we have two ways to create embeddings for each video

- Find word embeddings (from pre-trained word2vec model) for each word in a document and average it (Average of pre-trained Word embedding).
- Train a Doc2Vec model on the existing data and learn the document vector for each video

Note- It is always advised to train document vectors as pre-trained document vectors will be meaningless given they can be trained on some documents which are very different.

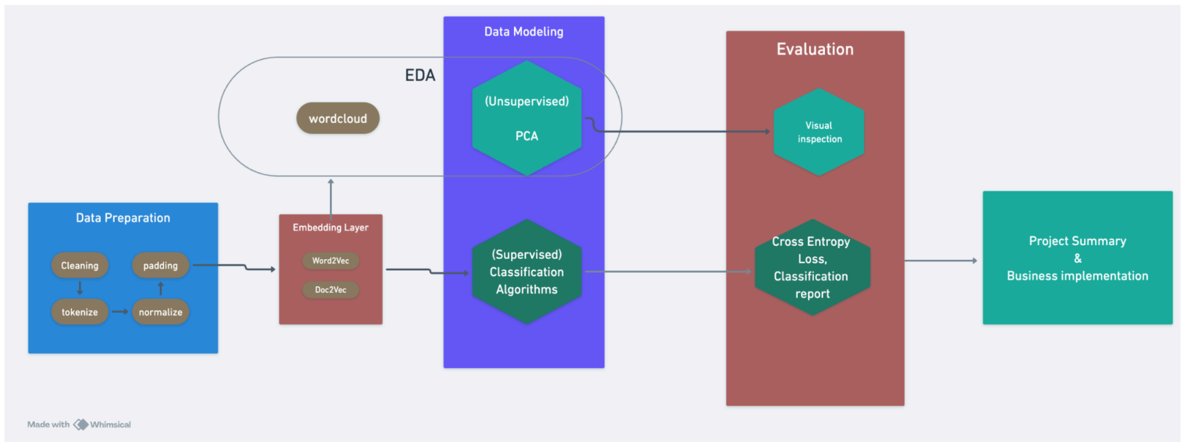
Discussion of how the firm / industry traditionally attempts to solve this problem

Youtube uses a sophisticated 2-stage recommendation process where the first stage is candidate gene ration where a few hundred videos are picked from a corpus of millions of videos. The learning problem is formulated as an extreme multi-class classification problem - out of all existing videos, predict the ones that the user engaged with. Then comes the ranking stage which further narrows down and sorts the candidates that end up in front of the user. In contrast, this project aims to analyze YouTube comments to explore the potential of incorporating comment text into YouTube's recommendation algorithm, a feature not currently utilized.

Dataset

The Dataset we picked was from our DDR project which consisted of 500 rows, each representing a video in 4 different genres i.e Lo-fi, Sports, Movie and Documentary. For each of the videos, all the comments from the comment section were used for analysis. There were about 457k tokens (words) on which this analysis is done. The final processed and cleaned data frame looked like the above. The videos were searched based on the keywords which were meticulously selected to maintain variation in the videos inside a genre.

Project Flow

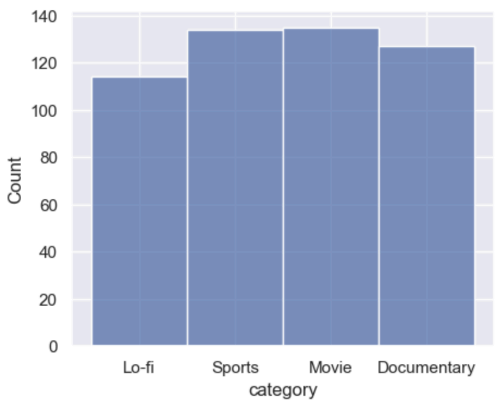


Above is a general project flow where we start off with data preparation where we go through the usual NLP data cleaning process which involves 4 critical steps - Stopword removal, stemming, lemmatization and tokenization. Post that we have two options of converting words into vectors - word2vec or doc2vec both of which we have already discussed above.

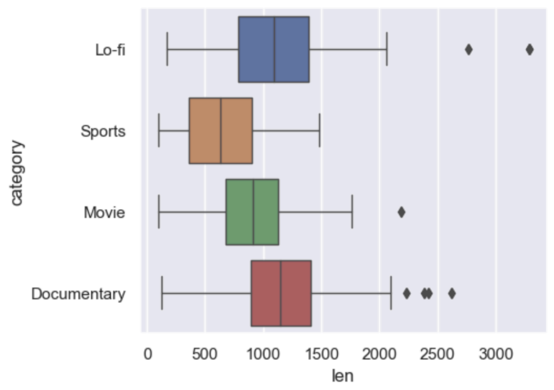
In short, the word2vec method in our case will convert each word into a 300 dimensional vector and average it for all words in that corpus to describe the document (in our context the document is the comments in a video). From our understanding, averaging word embedding is not a great way to describe a document. Doc2vec is another way to embed a corpus of words in a document and we will try to use both the ways and see how each of them perform in capturing the information in the comments sections.

Analysis

Before we start, we first look at number of videos whose comments are being analyzed.

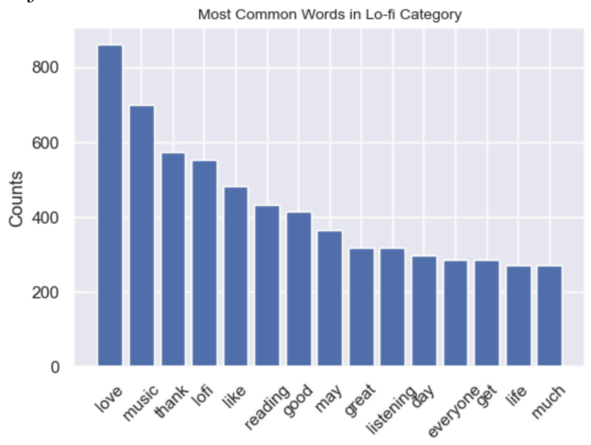


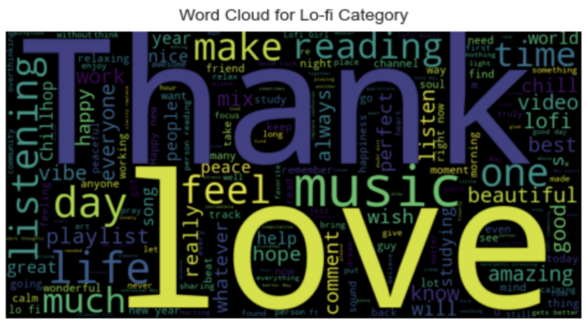
The above histogram plot shows that our dataset has an even distribution of videos for each genre showing that our dataset is evenly distributed. This would help us avoid any class imbalance problem that might have arised if we built any classification model. Next we will take a look at if we have enough comments (textual data) in our videos to properly describe it. The below chart represents how many words (or tokens) are present in each of our videos. Here we can see that most of our videos have more than 500 words which is enough to describe it in vectors.



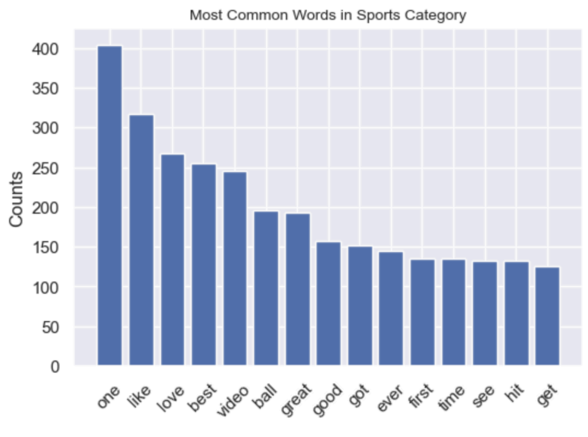
Now let's analyze each of the genres separately and look at the most commonly appearing words. This will give us a general idea of how each of the genres can be represented separately by the corpus of the comments.

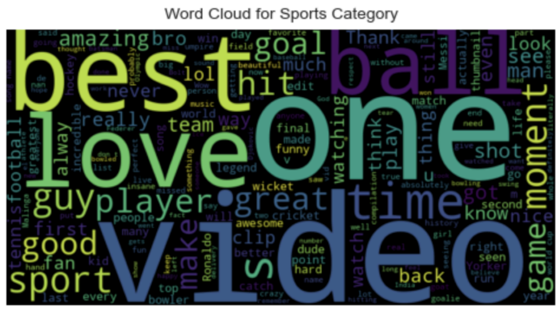
Lo-fi



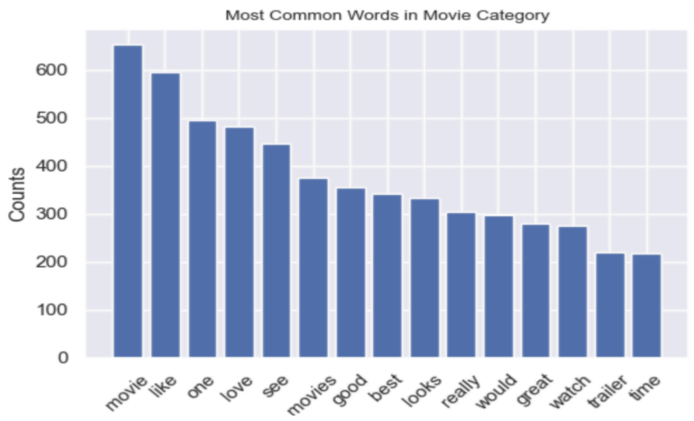


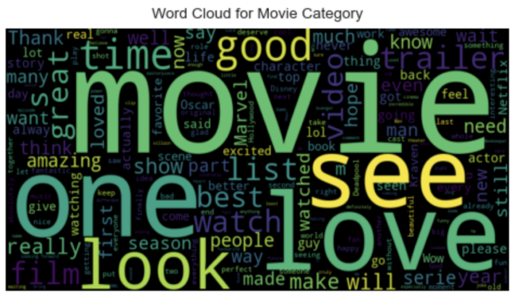
Sports



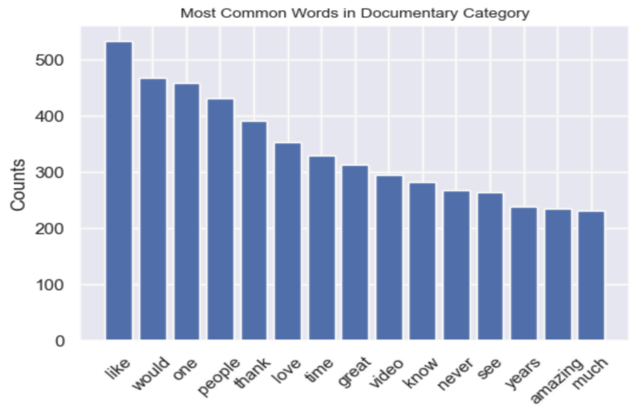


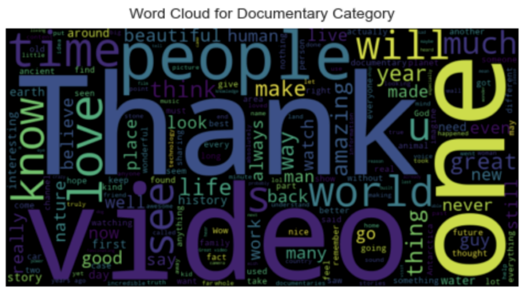
Movie





Documentary

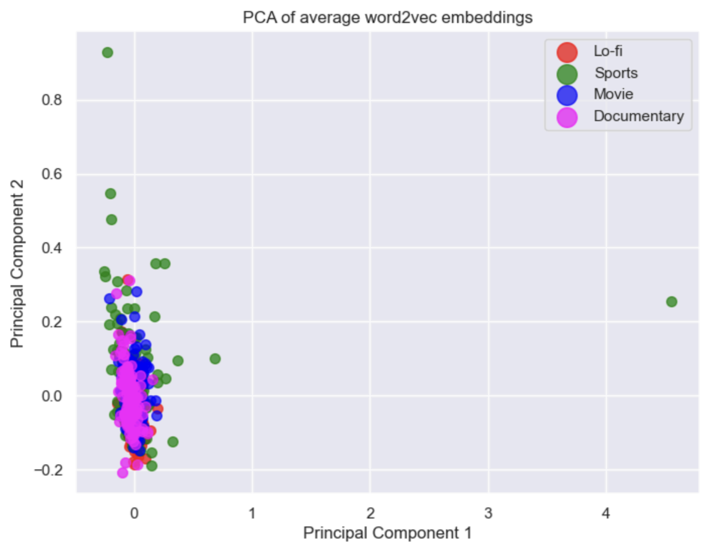




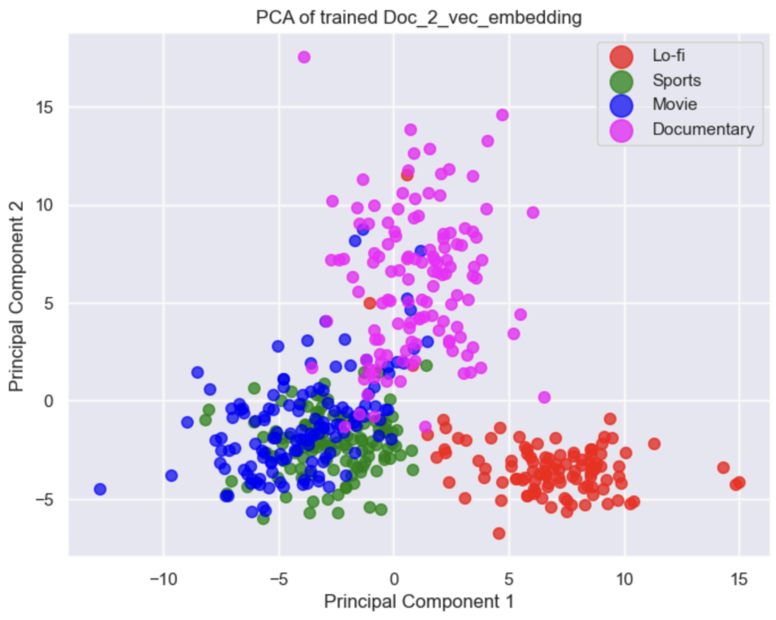
We choose to look at wordcloud and the word frequency plot, which gives us a fair idea of the top occurring words. Looking at the above plots, we can confidently say that our videos have sufficiently different words to be able to classify them.

PCA Analysis

PCA is commonly used as a method to visualize data and we will do the same to try and see how the videos align themselves on a scatter plot with just the first 2 components. Here each dot is a unique video which is described by all the comments it received and converted into a 300-dimension vector (but we are only visualizing the first 2 components here).

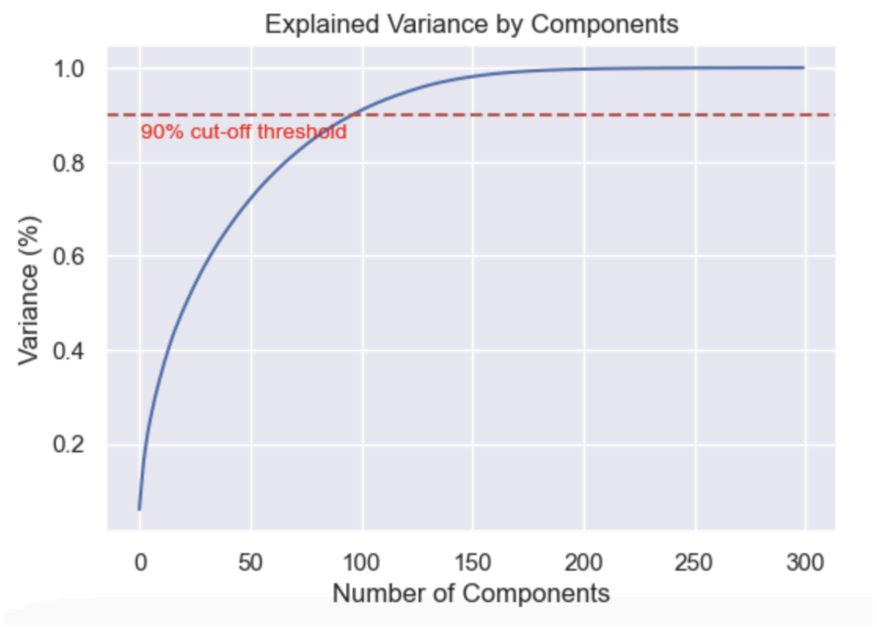


Next, we will try to see if training a new doc2vec embedding on our dataset will yield better results.



The averaging of pre-trained word2vec embedding doesn't seem to capture the difference between these different genres of videos as they all seem to be clustered together. The new document embedding for each video learned on the corpus of comments clearly shows that doc2vec does really well to separate videos of different genres. Here each dot is a unique video which is described by all the comments it received. So any classification algorithm we build will use the doc2vec embedding instead of averaging word2vec.

One clear problem we face here is the problem of overfitting. As our rows of data and features are pretty much the same, we might be overfitting if we try to build any multiclass classifier algorithm. So we will use a scree plot to see if we can drop some dimensions and reduce the number of features while retaining as much information as possible.



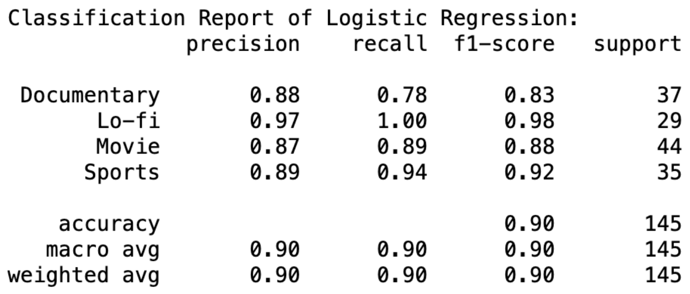
The above scree plot shows that we can keep about 100 first components after PCA while retaining 90% of the variance - this will reduce our model's performance but will massively help us get over the problem of overfitting, or the curse of higher dimension.

Models Used

Next we will use 3 separate algorithms to see if we can classify our videos just based on their comments (using the first 100 components from the doc2vec embedding which we trained on our dataset itself). We picked one classical ML algorithm and two tree based algorithms. Again, our dataset here is of 500 videos which we have split into 70-30 as train and test subset, where our algorithm would learn on 70% of the rows and be tested on the remaining 30% data.

Below are the classification reports of the 3 separate models.

- **Logistic Regression**:

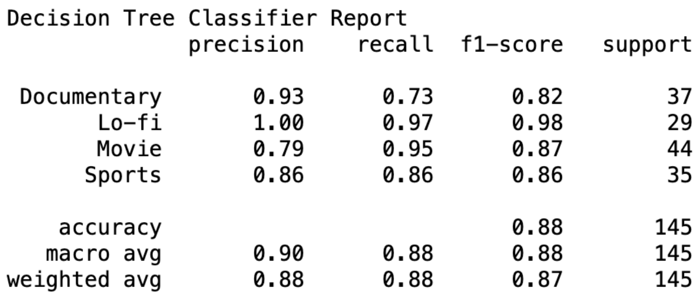


```
Classification Report of Logistic Regression:
              precision    recall  f1-score   support

 Documentary       0.88      0.78      0.83        37
       Lo-fi       0.97      1.00      0.98        29
       Movie       0.87      0.89      0.88        44
      Sports       0.89      0.94      0.92        35

    accuracy                           0.90       145
   macro avg       0.90      0.90      0.90       145
weighted avg       0.90      0.90      0.90       145
```

- **Decision Tree**:

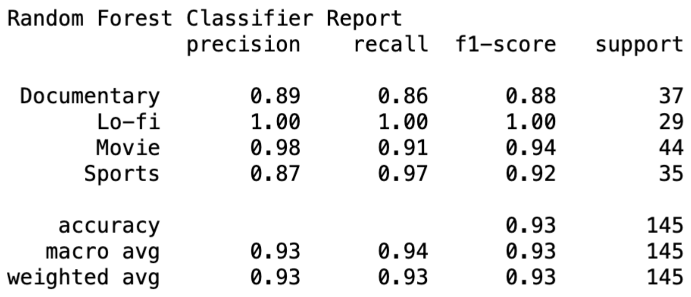


```
Decision Tree Classifier Report
              precision    recall  f1-score   support

 Documentary       0.93      0.73      0.82        37
       Lo-fi       1.00      0.97      0.98        29
       Movie       0.79      0.95      0.87        44
      Sports       0.86      0.86      0.86        35

    accuracy                           0.88       145
   macro avg       0.90      0.88      0.88       145
weighted avg       0.88      0.88      0.87       145
```

- **Random Forest**:



```
Random Forest Classifier Report
              precision    recall  f1-score   support

 Documentary       0.89      0.86      0.88        37
       Lo-fi       1.00      1.00      1.00        29
       Movie       0.98      0.91      0.94        44
      Sports       0.87      0.97      0.92        35

    accuracy                           0.93       145
   macro avg       0.93      0.94      0.93       145
weighted avg       0.93      0.93      0.93       145
```

Here if we look at the results, we are successfully able to classify the videos into their genres by a high degree of accuracy. A high F-1 score of 0.93 for random forest shows that comments are a good way to describe a video and its genres. Another interesting thing we noticed is that the Lo-fi genre is almost perfectly identified by our algorithms, something we also noticed in our PCA plot.

Recommendations and Business Values

Given how our algorithm performed, our recommendation will be for Youtube to explore using textual data in their comments section in their recommendation algorithm.

Summary and Conclusions

The project's findings were promising, with classification algorithms, particularly the Random Forest model, demonstrating high accuracy in genre classification based on comments. This success underscores the untapped value within YouTube's comment sections for enhancing content recommendations. The conclusion drawn from the project is that incorporating textual analysis of comments could significantly improve the accuracy and relevance of YouTube's recommendation algorithm. Therefore, the recommendation for YouTube is to consider integrating comment-based textual data into their recommendation processes, potentially leading to more personalized and engaging user experiences.