HACETTEPE UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ARTIFICIAL INTELLIGENCE ENGINEERING

CommNet: Sentiment Analysis and Video Keyword Prediction from YouTube Comments

Final Project Report

AIN420 - INTRODUCTION TO DEEP LEARNING

Asst. Prof. Dr. Cemil ZALLUHOĞLU

Hüseyin Eren DOĞAN – 2210765009

Tanay TEKÖN – 2200765013

2024-06-09

Abstract

Context: **Exploring YouTube comments with deep learning approaches to analyze and predict viewer sentiment and video keywords.**

Purpose: **To develop an AI model that generates contextually relevant comments, analyzes sentiment (positive, negative, or neutral), and predicts video keywords, providing content creators with valuable insights into viewer engagement.**

Method: **Recurrent Neural Networks (RNNs) and Natural Language Processing (NLP) techniques are employed for comment generation and sentiment analysis. Artificial Neural Networks (ANNs) are utilized for sentiment and keyword prediction, leveraging pre-trained embeddings and custom models.**

Results: **The AI model effectively analyzes comment sentiment, achieving 81% accuracy. Keyword prediction, while challenged by the diversity of YouTube content, demonstrates promising results. The project's initial goal of generating comments using generative AI was not fully realized due to limitations in resources and time.**

1. Introduction:

In the rapidly changing digital media space, YouTube has established itself as a leading platform for engaging with video content and interacting within a community. With billions of users interacting on a daily basis, the comment sections of YouTube videos have grown into a vital environment for viewer expression and interaction. These comments shed light on the sentiments and levels of interaction of the audience, offering content creators feedback that might guide the improvement of content and relationship management with their audience.

As rich as the information carried in YouTube comments is, there are still several substantial challenges when trying to mine and analyze it. A lot of the work is manual and is, therefore, time-consuming, not to mention the large number of comments available. Secondly, and not least, is the heterogeneity and variability in the language, tone, and context of comments. Our project, earlier in inception, had focused on these issues with the hope of providing a solution to users through the utilization of generative AI. However, this method proved to be less efficient than desired, yielding generated comments of lesser quality and relevance.

In moving away from the original idea, the project, CommNet, will focus on the sentiment analysis and keyword prediction to unlock valuable insight locked within YouTube comments. The goal is to develop a strong and scalable AI model for these tasks in order to automate and scale up the understanding of viewer engagement. An approach was proposed that integrates techniques for both sentiment analysis and keyword prediction, such as training Artificial Neural Networks (ANNs) and Long Short-Term Memories (LSTMs), and using Natural Language Processing (NLP) techniques to infer the sentiment of comments: classifying it into positive, negative, or neutral tone. The keyword analysis and prediction component identify prevalent themes and topics in the comments, hence providing actionable insights to content creators.

This new approach will automate the uncovering of meaning in comments, allowing content creators to derive deeper insights into the sentiment and prevalent topics that will most likely resonate with their audiences. In the end, CommNet will offer end-to-end insights of viewer engagement and sentiment trends to content creators, helping them fine-tune their strategies to create a more connected audience base.

2. Background & Related Work:

Table 1: Summary of the related work

Study	Features	Classifiers	Performance	Datasets
Mehta & Deshmukh (2022)	**Views, comments, likes, dislikes, duration, published date, category**	**LinearRegression, SVM, Decision Tree, Random Forest, ANN**	**RMSE: 260.193 (Decision Tree)**	**Kaggle dataset (15k YouTube videos)**

This paper proposes using machine learning and deep learning models to predict YouTube ad view sentiment. The authors use linear regression, support vector machines, decision trees, random forests, and artificial neural networks to train their models on a dataset of 15,000 YouTube videos. They find the decision tree model to perform the best, with a root mean square error (RMSE) of 260.193. The authors conclude that their approach can be employed to effectively analyze large amounts of data and improve the sentiment analysis efficiency.

The contributions of this project build upon their study, focusing on the comment-level analysis and incorporating sentiment and keyword prediction to gain finer-grained insights into viewer engagement. It has used advanced NLP techniques to categorize comments into positive, negative, or neutral sentiments and introduced keyword prediction to identify trending topics within comments. This not only enriches the feature set but also helps content creators with actionable insights. Even though the initial attempt at employing generative AI for comment generation was not successful, it was fruitful for the project because it shifted the focus to more effective sentiment analysis and keyword prediction. Comprehensive reports that provide viewer engagement metrics and their sentiment trends are being delivered in order to move the field forward beyond the Mehta and Deshmukh foundational approach.

3. Methodology:

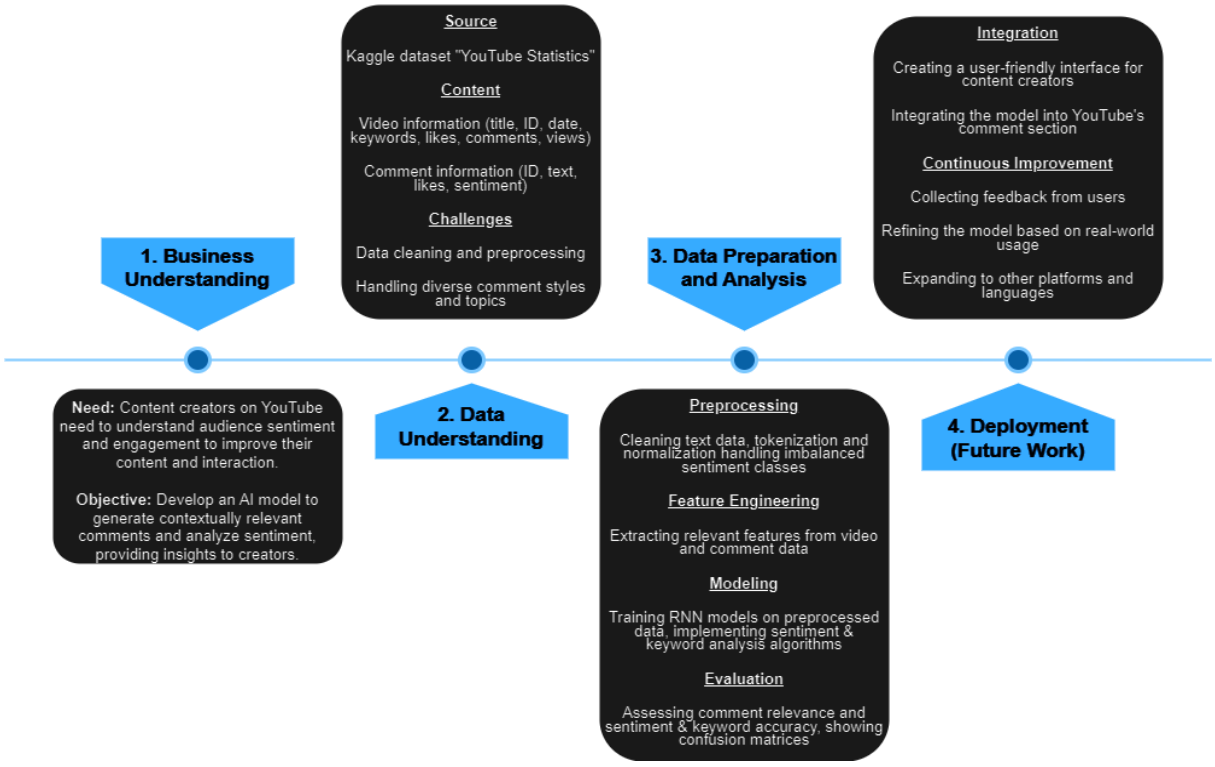


Figure 1. Methodology Diagram

3.1. Data Exploration and Understanding

[2] Video-Stats and Comments datasets are used. Video-Stats contains basic information about 1881 different. The attributes of the video statistic dataset are: video ID, number of likes video has, number of comments video has, number of views video has, number of views video has, the publishing date of the video, keyword which specifies the topic of the video. For each video in the video statistics file, comments data contains the top ten most relevant comments. It contains 18409 different entries. The attributes of the comments dataset are video ID, number of likes the comment has, comment text, and sentiment of the comment (0: Negative, 1: Neutral, 2: Positive). These two datasets are combined based on the video ID column. Figure 2 shows the distribution of video keywords associated with each video and Figure 3 depicts the most used words in the comments.

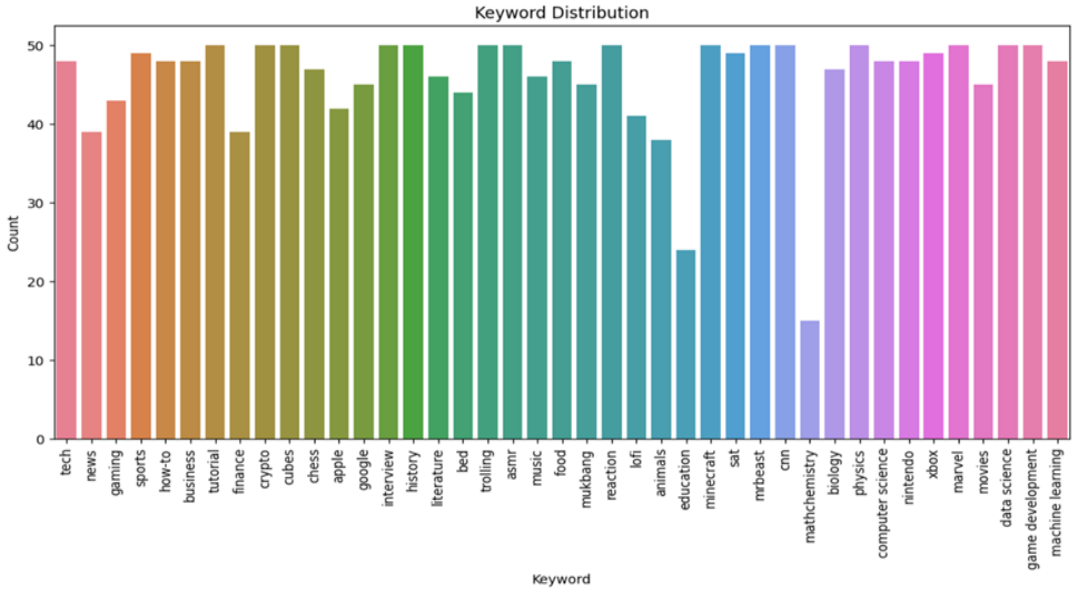


Figure 2. Distribution of Video Keywords

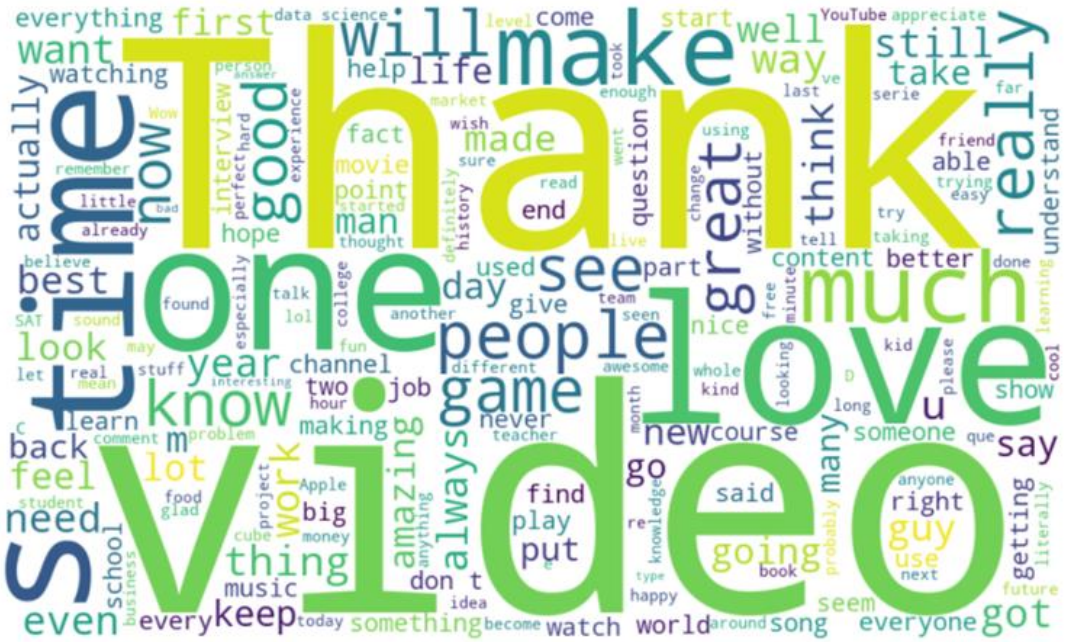


Figure 3. Most commented words in the comments data

In addition, the distribution of sentiments is visualized in Figure 4. There is a significant class imbalance in sentiments. Some sampling techniques like random oversampling to the train set are applied to solve this issue.

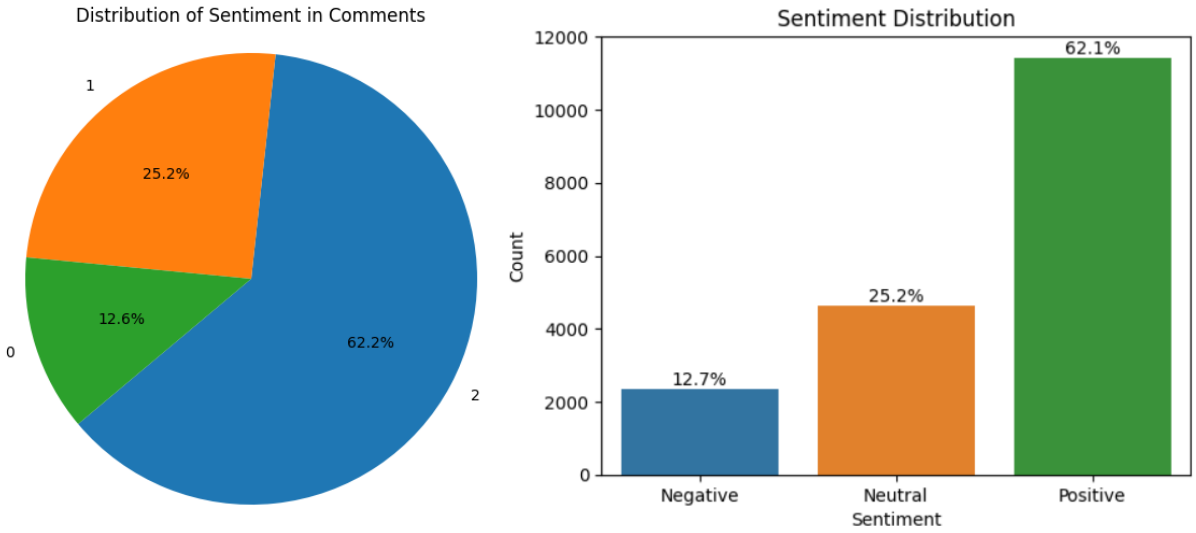
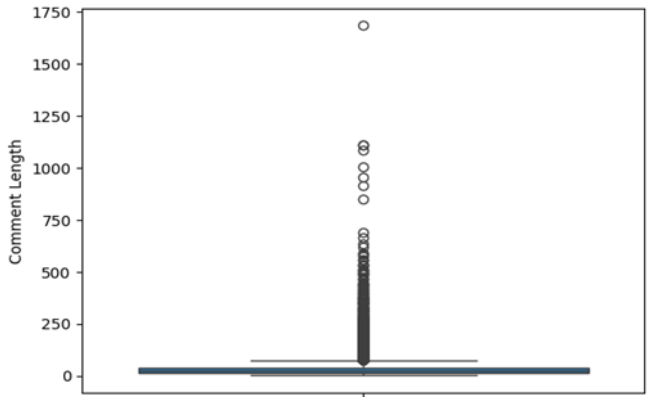


Figure 4. Sentiments Distribution

3.2. Data Preprocessing

As a data preprocessing technique, an outlier analysis is performed using the box plots to look at the length of the comments and put them into an appreciable range while removing the outliers. The box blot graph which visualizes the outliers on comment length and quartile values are depicted in Figure 5. Also, the distribution of the comment length before and after removing outliers that have comments with a length bigger than 75 is shown in Figure 6.



1st quartile	13
3rd quartile	38
IQR	25

Minimum (Q1 - 1.5 * IQR)	1
Maximum (Q3 + 1.5 * IQR)	75

Figure 5. Outliers on Comment Length

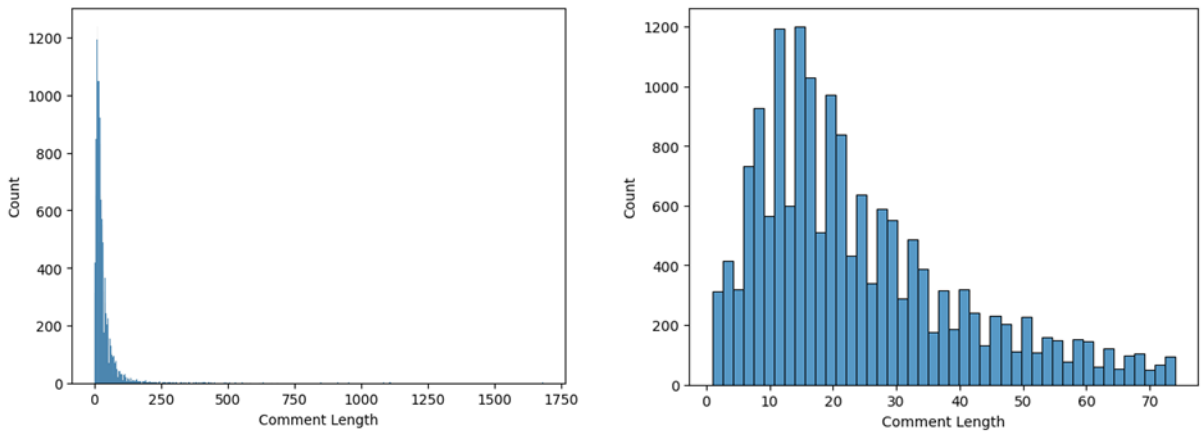


Figure 6. Distributions of Comment Length before and after removing outliers

Then emojis, unnecessary white spaces, links and mentions, non UTF-8/ASCII characters and signs, punctuations, hashtags, and stop words are removed. Lemmatization is applied. Lemmatization is not applied when using the USE encoder. Lastly, the train-validation-test split is made in balance.

3.3. Approach

3.3.1. Naive Bayes with TF-IDF Vectorizer

TF-IDF Vectorizer computes the TF-IDF score for each word in each comment. It considers how common the word is in this specific comment and how common it is in all the other comments, therefore considering both the word's frequency within a comment (term frequency) and its overall prevalence across all comments (inverse document frequency). Words that appear frequently across all comments will then get a much lower weight compared to words that are specific to a particular class of sentiment.

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Equation 1. TF-IDF Vectorizer Calculation

A MultinomialNB classifier from scikit-learn was selected for sentiment classification. This is a variant of Naive Bayes and thus well-suited to dealing with discrete features such as word counts in this case, where that is the way the comments are represented by TF-IDF.

The TF-IDF vectors calculated in the prior step were utilized to train the Naive Bayes model. The model learned the relationships between these TF-IDF features and the corresponding sentiment labels (positive, negative, and neutral) assigned to the comments during preprocessing.

However, the assumption of independence between features in Naive Bayes might not perfectly hold true for language data where words can be semantically related. While TF-IDF is a common choice, recent research suggests pre-trained word embeddings might outperform it for sentiment analysis, particularly with more complex models.

3.3.2. BERT (Bidirectional Encoder Representations from Transformers)

BERT is a powerful pre-trained deep learning model based on the Transformer architecture, with its architecture designed for a variety of Natural Language Processing (NLP) tasks. Unlike traditional word embedding models that consider the context of a word based on its surrounding words (limited window), BERT is bidirectional in nature. It processes the whole sentence at once, in such a way that it learns both preceding and subsequent words for any given word to understand its meaning. These allow BERT to grasp complex interrelations between words and further have a deep understanding of the general sentiment within a sentence.

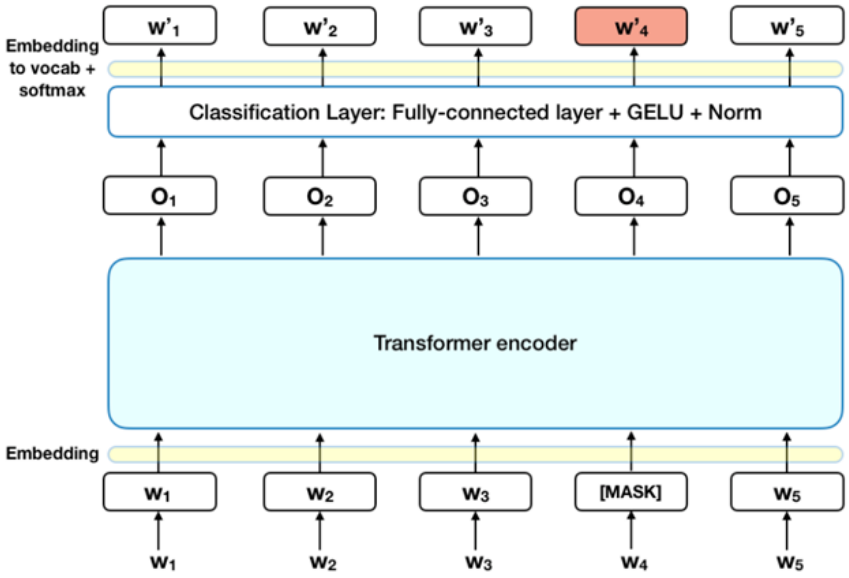


Figure 7. BERT Architecture

While BERT is pre-trained on a huge dataset of plain text and text pairs, it can still be fine-tuned for the user's specific task—in this case, sentiment classification. The fine-tuning process occurs in the following way:

Pre-trained BERT Model: A pre-trained BERT model (bert-base-uncased from the Hugging Face Transformers library) was loaded. This model already possesses the general knowledge of language learned from its pre-training phase.

Adding a Sentiment Classification Layer: A layer consisting of neurons was added on top of the pre-trained BERT model. This layer acts as a classifier, taking the output from BERT, which is the sentence embedding, and predicting the sentiment category—positive, negative, or neutral.

Fine-tuning with Labeled Data: A pre-trained BERT model with an added classification layer was fine-tuned on the sentiment-labeled YouTube comments dataset. During the fine-tuning process, the model learns the weights of its internal layers to give more importance to features of the data important for sentiment classification in YouTube comments. BERT models are computationally expensive to train and run. Due to this reason, only the last layer of this model was trained.

3.3.3. Word2Vec Google News 300 Word Embedding

Word2Vec Google News 300 is a kind of word embedding, a method in Natural Language Processing to numerically represent words. Word2Vec is an algorithm used to create the word embeddings. Word2Vec is a popular method that estimates the relationship between two words by training it over a large text corpus and predicting surrounding words based on the input word. The intuition is that words that are functionally similar should have related vector representations.

Word embeddings are trained on data derived from Google News. In other words, it trained on a corpus primarily consisting of news articles, and thus, probably contains lots of examples of informal language, slang, and current events terminology. This word embedding model has 3 million vocabulary size. Each word is represented by a vector containing 300 numbers. This would imply a high dimensionality, allowing for the relationships between the words to be more complex but needing more computational resources.

This is trained on news articles and, consequently, informal language, slang, and the sayings of the moment, likely to make it well-suited to the task of understanding comments on YouTube.

3.3.4. Glove Wiki Gigaword 50

GLOVE-WIKI-GIGAWORD-50 is a kind of word embedding used in Natural Language Processing. GLOVE stands for Global Vectors for word representation. It is an algorithm used in creating word embeddings by way of aggregating the statistical occurrence of words in a large text corpus. Words that frequently appear together are considered semantically similar and given closer vectors in the embedding space.

These word embeddings were trained on the Wiki-Gigaword dataset. This is Wikipedia text, which is likely to cover a wide range of topics and vocabulary, possibly including formal language and academic terms. Gigaword is a large corpus of news articles, which provide more extensive vocabulary than just Wikipedia text. Each word is represented by a 50-number vector. Lower dimensionality is more computationally efficient but might grasp fewer complex relationships between words.

These embeddings, trained on both the Wikipedia text and the Gigaword, are likely to capture a much greater range of vocabulary, both in topics and in the forms of language—something that proves to be particularly helpful for text data analysis when the text data is from diverse domains. Due to the incorporation of the Wikipedia text, these embeddings might be more biased towards formal language and academic vocabulary in comparison with Word2Vec Google News 300.

3.3.5. Universal Sentence Encoder (USE)

Universal Sentence Encoder (USE) is a pre-trained deep learning model developed by Google AI for NLP tasks. It differs from the traditional word embedding models, which encode individual words, with USE focusing on encoding whole sentences into dense numerical vectors. These numerical vectors retain the meaning and sentiment of the sentence.

The key advantage of the model is its pre-training on a very large dataset of text and text pairs. Such pre-training forms an effective basis for the development of a model concerning numerous NLP tasks, including sentiment analysis and keyword extraction.

A significant advantage of USE is that it encodes complete sentences, and therefore, it has the capability to capture the overall sentiment and meaning of a sentence, hence possibly providing better results in sentiment analysis than models that are based on word-level features.

Taken together, this makes the Universal Sentence Encoder a versatile tool for sentiment analysis and keyword extraction tasks. Due to its pre-trained nature and efficiency, the tool is invaluable in general NLP applications.

3.3.6. LSTM (Long Short-Term Memory)

Traditional neural networks cannot capture long-term dependencies within sequences such as text data. Long Short-Term Memory networks are a type of recurrent neural network architecture that deals with that. LSTMs have a memory cell that enables them to take information about all the previous elements in a sequence and use it for understanding the current element. This makes them well-equipped for tasks where the order of words and the context in which they are used is important, such as for sentence sentiment analysis.

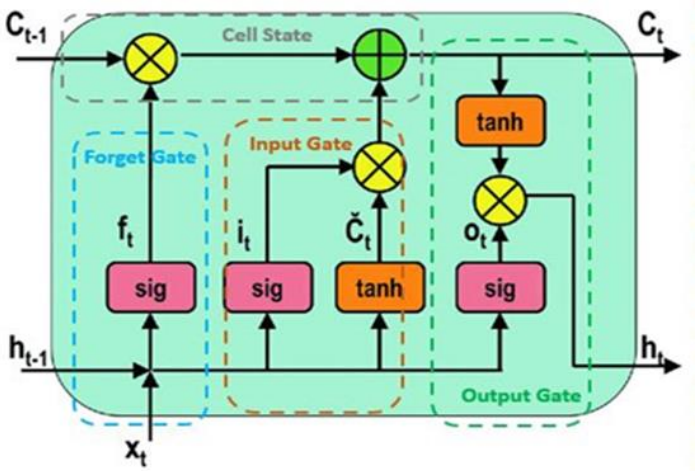


Figure 8. LSTM Cell

Here, the preprocessed comments are being represented in sequence word embedding vectors going into the input layer. Then the LSTM layer(s) processes the sequence of word vectors. The memory cells of LSTMs capture information about preceding words and how they are associated in order to make a decision about the overall sentiment conveyed by that comment. The final layer predicts the comment sentiment category (positive, negative, neutral) by using a SoftMax activation function.

3.3.7. Bidirectional LSTM

Bidirectional LSTMs are an extension of LSTMs that address a potential limitation. Bidirectional LSTMs are designed to process sequences of data in two directions: forward and backward. The Bidirectional LSTM should be able to capture not only the context of preceding words but also the influence of following words on the sentiment of a comment.

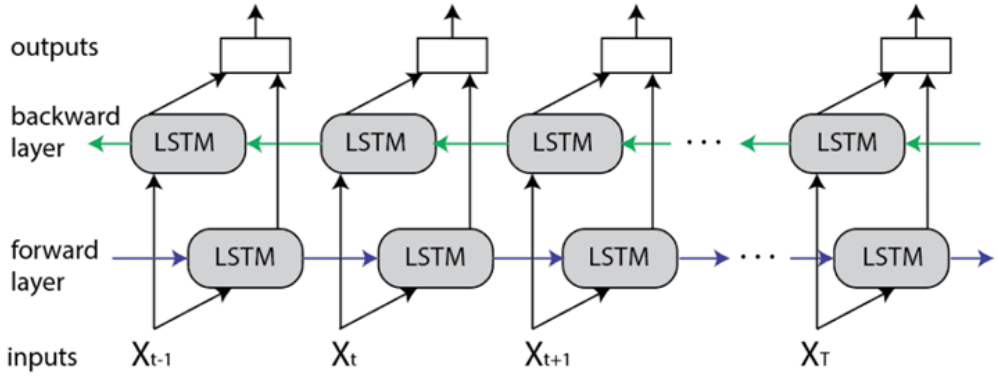


Figure 9. Bidirectional LSTM Architecture

Bidirectional LSTM networks follow the same design of a standard LSTM network; the difference arises in the implementation of two different LSTMs:

Forward LSTM: This LSTM processes the sequence of word embeddings in the forward direction, capturing the influence of preceding words.

Backward LSTM: This LSTM processes the same sequence in the reverse direction, capturing the influence of following words.

Then, the output states from both LSTMs are merged and passed through an output layer.

3.4. Algorithm Selection and Model Design

Different word embedding models have been added to the models.

Sentiment Analysis:

- BERT Tokenizer —> BERT for Sequence Classification
- TF-IDF —> Multinomial Naive-Bayes
- Glove-Wiki Gigaword-50 —> ANN Model 1&4
- Word2Vec-Google-News-300 —> ANN Model 2
- Custom Embedding —> ANN Model 3
- Universal Sentence Encoder —> ANN Model 5&6

Comment Generation:

- GPT-2 Tokenizer —> GPT-2

Keyword Prediction:

- Universal Sentence Encoder —> ANN Model 7&8

ANN model 5 and 7, and 6 and 8 have same architecture except the last SoftMax layer, since number of classes are different for sentiment analysis and keyword prediction. Here are archictectures of the ANN models:

ANN MODEL 1:
InputLayer(glove-wiki-gigaword-50 embedding),
Dense(256)ReLU, Dropout(0.1),
Dense(512)ReLU, Dropout(0.1),
Dense(512)ReLU, Dropout(0.1),
Dense(256)ReLU, Dropout(0.1),
Dense(3)SoftMax

ANN MODEL 2:
InputLayer(word2vec-google-news-300 embedding),
Dense(256)ReLU, Dropout(0.1),
Dense(512)ReLU, Dropout(0.1),
Dense(512)ReLU, Dropout(0.1),
Dense(256)ReLU, Dropout(0.1),
Dense(3)SoftMax

ANN MODEL 3:
InputLayer(custom-embedding-200),
LSTM(64),
Dropout(0.1),
LSTM(64),
Dense(32)ReLU [L1(0.1) L2(0.1)],
Dense(3) SoftMax

ANN MODEL 4:
InputLayer(glove-wiki-gigaword-50 embedding),
LSTM(32),
LSTM(16),
Dense(3) SoftMax

ANN MODEL 5:
InputLayer(USE Layer),
Dense(512) ReLU,
Dense(3) SoftMax

ANN MODEL 6:
InputLayer(USE Layer),
Bidirectional LSTM(128),
Bidirectional LSTM(64),
Dense(3) SoftMax

ANN MODEL 7:
InputLayer(USE Layer),
Dense(512) ReLU,
Dense(41) SoftMax

ANN MODEL 8:
InputLayer(USE Layer),
Bidirectional LSTM(128),
Bidirectional LSTM(64),
Dense(41) SoftMax

4. Experiments

4.1. Comment Generation with GPT-2

Initially, the objective was to generate comments that reflected the features and content of the videos. However, it was quickly realized that this was a more challenging task than anticipated due to the complexity of capturing video content and translating it into coherent textual comments.

Despite these challenges, the experimentation of a pre-trained GPT-2 model continued. The pre-trained parameters were fixed; only the final layers of the model were trained. Then, the model was tested by generating comments conditioned on some prompt, which yielded results that were not quite coherent enough. Examples of generated sequences are shown below:

Generated Sequence 1:
"I hope that the the a the an a the athe the aa The a an A aThe IThea a TheaTheA THE_S TH N R M S,M"
Generated Sequence 2:
"I hope that the thethethe the, theThesons of thes,The the Theson theIth."
Generated Sequence 3:
"I hope that $s."
Generated Sequence 4:
"The sine of the, the sinity ofthe, sinisineofthe."

The generated sequences demonstrate that the model struggled to produce coherent and meaningful comments. Freezing the pre-trained parameters and only training the final layers might have limited the model's ability to adapt to the specific task of comment generation. Even though GPT-2 is indeed capable, more advanced fine-tuning techniques or larger datasets could be used to generate high-quality comments that are relevant to video content.

4.2. Sentiment Analysis with BERT

Initially, sequence classification was attempted using BERT, a widely used and popular pre-trained language model. Upon loading the model's parameters, it was found to contain 109,486,103 parameters, with each epoch requiring approximately 5-6 hours to complete. Consequently, a decision was made to freeze all layers except the last one and to provide only a portion of the data to the model, resulting in a reduced runtime of around 8-9 minutes per epoch.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_word_ids (InputLayer   [(None, 128)]             0         []
)

input_mask (InputLayer)      [(None, 128)]             0         []

segment_ids (InputLayer)     [(None, 128)]             0         []

tf_bert_for_sequence_class   TFSequenceClassifierOutput  109486085  ['input_word_ids[0][0]',
ification_1 (TFBertForSequ   (loss=None, logits=(None,              'input_mask[0][0]',
enceClassification)          5),                                    'segment_ids[0][0]']
                              hidden_states=None, atten
                             tions=None)

flatten_1 (Flatten)          (None, 5)                 0         ['tf_bert_for_sequence_classif
                                                                  ication_1[0][0]']

dense_1 (Dense)              (None, 3)                 18        ['flatten_1[0][0]']

==================================================================================================
Total params: 109486103 (417.66 MB)
Trainable params: 3863 (15.09 KB)
Non-trainable params: 109482240 (417.64 MB)
```

Figure 10. Architecture of Fine-tuned BERT Model

Adam optimizer with learning rate 0.00001 was used. As loss function, sparse categorical cross entropy was used. The model was trained for 20 epochs. Batch size 16 was used. Early stopping was used based on validation loss, with patience of 3 epochs.

The model's performance was not satisfactory, achieving only 58% validation accuracy and 62% accuracy on training and testing, incorrectly predicting all comments as positive. Subsequently, alternative approaches were explored to address this problem.

4.3. Sentiment Analysis with Naive-Bayes

Subsequently, another approach was tried that aimed to harness the probabilistic framework of Naive Bayes for making sentiment label predictions of high accuracy for new or unseen text data. MultinomialNB is used to estimate the probability of each sentiment class, given the observed word frequencies in the text. The method is simple and effective for carrying out sentiment analysis. TF-IDF was used to vectorize and transform the text data, followed by the use of GridSearch to tune the alpha parameter. Alpha values "0.0001, 0.001, 0.005, 0.1, 0.15, 0.2, 0.5, 1.0, 1.5, 2.0" are tested and the best alpha value was determined as 0.1.

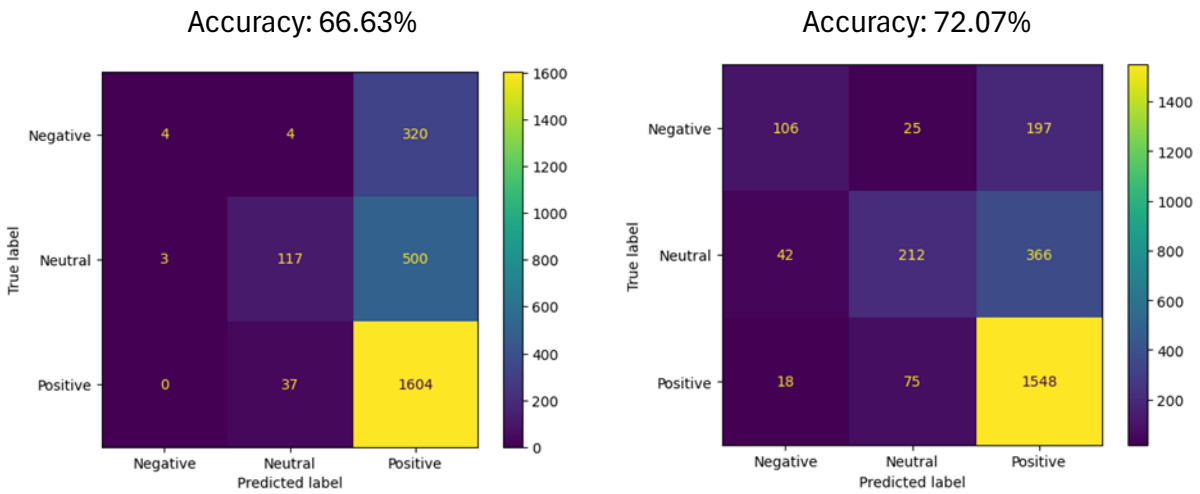


Figure 11. Naive-Bayes results before and after hyperparameter tuning

4.4. Artificial Neural Networks (ANNs)

4.4.1. Comment Sentiment Prediction

Another approach was pursued by implementing various word embedding models in neural networks, conducting a series of experiments with each model.

MODEL 1 (glove-wiki-gigaword-50)

In this model glove-wiki-gigaword-50 embeddings used with dense layers. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32. Early stopping based on validation accuracy with patience of 10 epochs was used. Training was stopped at 16th epoch, due to the early stopping.

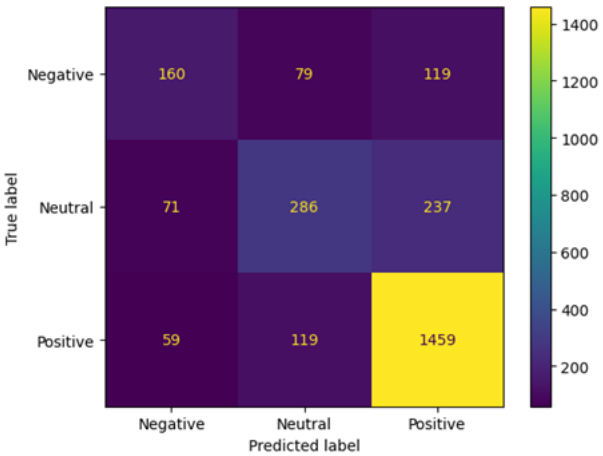


Figure 12. Confusion Matrix of Model 1 Results

Training Accuracy: 82.98% Validation Accuracy: 71.59% Testing Accuracy: **71.80%**

While the model's performance was not considered poor, exploration of a different word embedding model was undertaken to achieve further improvement.

MODEL 2 (word2vec-google-news-300)

In this model word2vec-google-news-300 embeddings used with dense layers. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32. Early stopping based on validation loss with patience of 10 epochs was used. Training was stopped at 13th epoch, due to the early stopping.

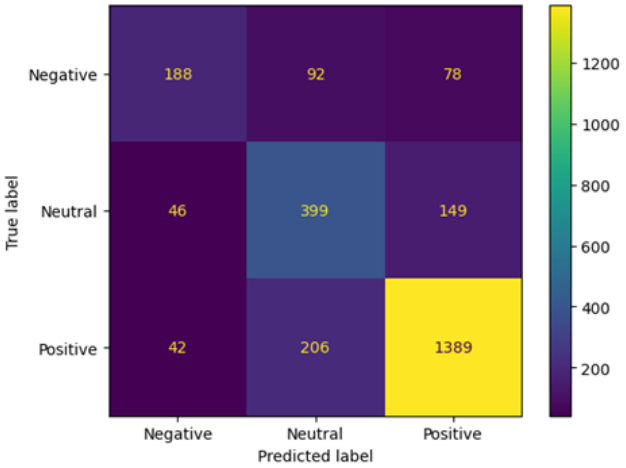


Figure 13. Confusion Matrix of Model 2 Results

Training Accuracy: 91.09% Validation Accuracy: 76.50% Testing Accuracy: **75.24%**

After observing an improvement with the different word embedding model, an investigation was initiated to determine whether a custom embedding layer could capture contexts potentially missed by pre-trained models.

MODEL 3 (custom embedding layer)

In this model custom embedding layer, with embedding dimension 200, used with LSTMs. Adam optimizer with learning rate 0.01 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32. Early stopping based on validation accuracy with patience of 5 epochs was used. Training was stopped at 6th epoch, due to the early stopping.

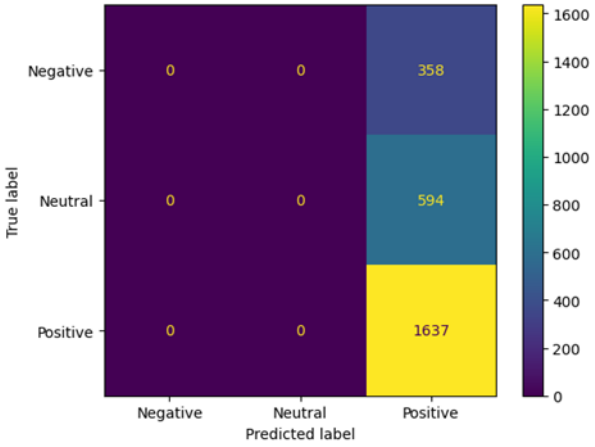


Figure 14. Confusion Matrix of Model 3 Results

Training Accuracy: 63.61% Validation Accuracy: 61.99% Testing Accuracy: **63.22%**

The model with a custom embedding layer clearly failed to capture any meaningful context from the data, as it predicted all comments as positive.

MODEL 4 (glove-wiki-gigaword-50) (with LSTM)

In this model glove-wiki-gigaword-50 used with LSTMs. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32. Early stopping based on validation accuracy with patience of 10 epochs was used.

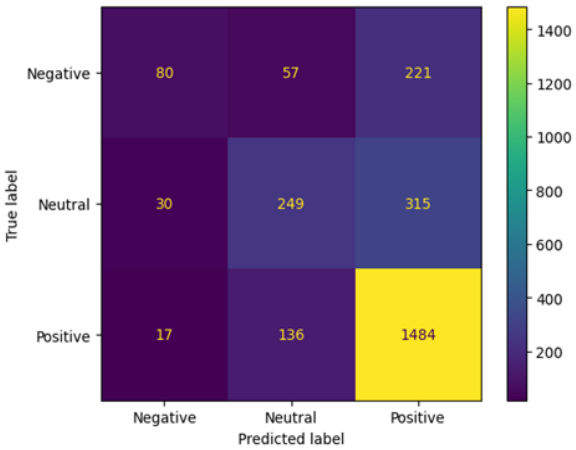


Figure 15. Confusion Matrix of Model 4 Results

Training Accuracy: 69.29% Validation Accuracy: 67.64% Testing Accuracy: **63.53%**

Subsequently, investigated whether using LSTM would improve the models by capturing the sequential relationships between words and the context of the comments. However, this approach did not yield significant improvements.

MODEL 5 (google-use)

In this model Google's USE was used with dense layers. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32.

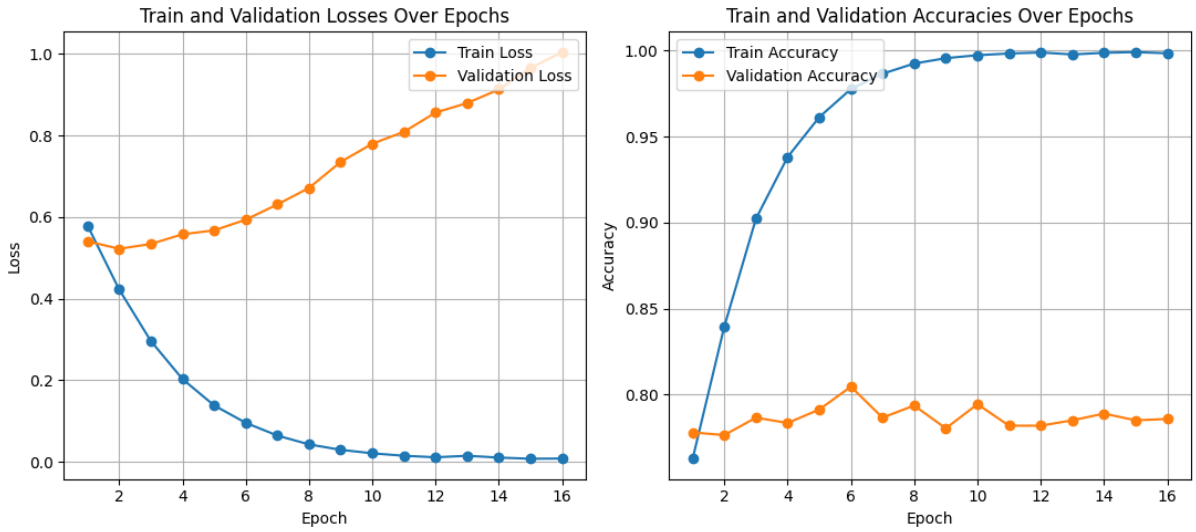


Figure 16. Train & Validation Losses and Accuracies Over Epochs

According to above figure, this model was overfitting. However, early stopping mechanism based on validation accuracy was used with patience of 10 epochs. In this case, the weights at the 16th epoch are used.

Training Accuracy:	97.76%
Validation Accuracy:	80.45%
Testing Accuracy:	**81.31%**

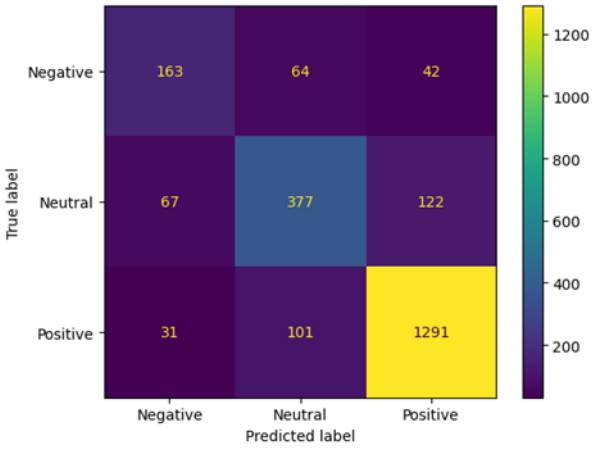


Figure 17. Confusion Matrix of Model 5 Results

Google's Universal Sentence Encoder proved to be more effective on this dataset. By using it as the first layer of the model and reducing the model's complexity to handle overfitting, achieved a notable improvement in performance.

Table 2: Classification results for Model 5

	precision	recall	F1-score	support
0	0.59	0.62	0.61	269
1	0.75	0.62	0.68	566
2	0.88	0.92	0.90	1423
accuracy			0.81	2258
macro avg	0.74	0.72	0.73	2258
micro avg	0.81	0.81	0.81	2258

MODEL 6 (google-use) (with LSTM)

In this model Google's USE was used with LSTMs. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32.

According to figure below, similarly with previous one, this model was also overfitting. However, early stopping mechanism based on validation accuracy was used with patience of 10 epochs. In this case, the weights at the 16th epoch are used.

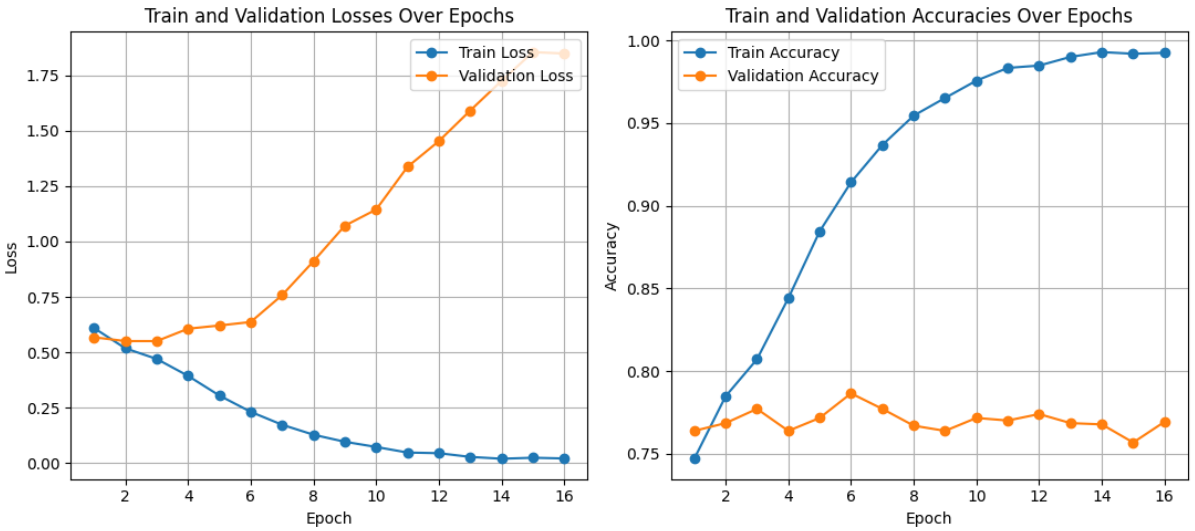


Figure 18. Train & Validation Losses and Accuracies Over Epochs

Training Accuracy:	91.42%
Validation Accuracy:	78.66%
Testing Accuracy:	**80.15%**

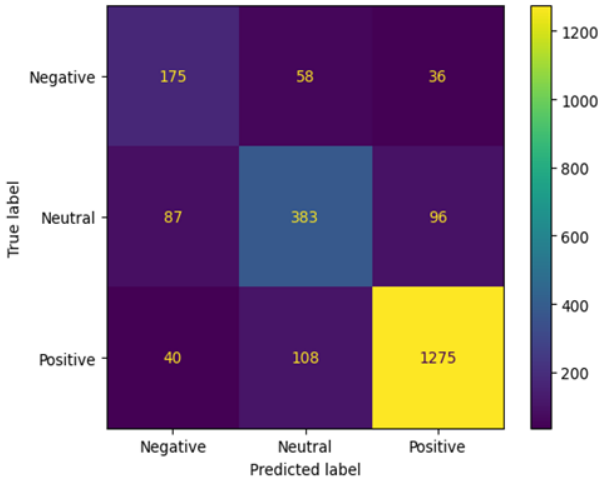


Figure 19. Confusion Matrix of Model 5 Results

While the overall performance metrics remain unchanged, the addition of LSTM to the model failed to enhance the performance. However, this task highly depends on the processing of sequences and context in comments. Generally, the LSTM model had a slightly lower performance on overall metrics; it achieved better accuracy among the neutral and negative classes, which is an improvement since most of the data contain positive comments. This makes LSTMs more efficient for our particular needs.

Table 3: Classification results for Model 6

	precision	recall	F1-score	support
0	0.54	0.64	0.59	269
1	0.68	0.67	0.68	566
2	0.91	0.89	0.90	1423
accuracy			0.80	2258
macro avg	0.71	0.73	0.72	2258
micro avg	0.81	0.80	0.80	2258

4.4.2. Keyword Prediction

MODEL 7 (google-use)

In this model Google's USE was used with dense layers. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32.

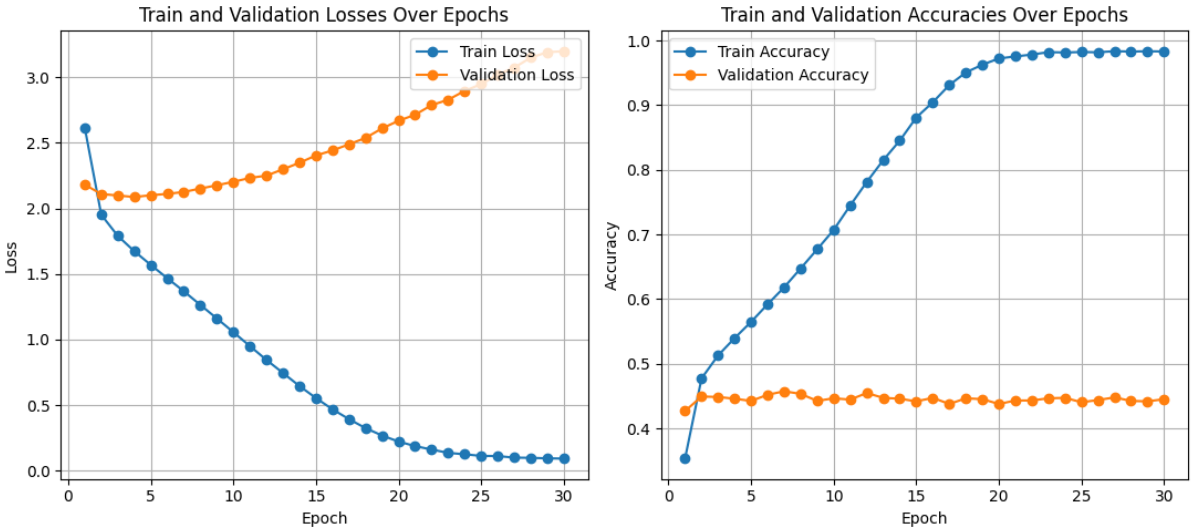


Figure 20. Train & Validation Losses and Accuracies Over Epochs

Early stopping mechanism based on validation accuracy was used with patience of 30 epochs. In this case, the weights at the 7th epoch are used.

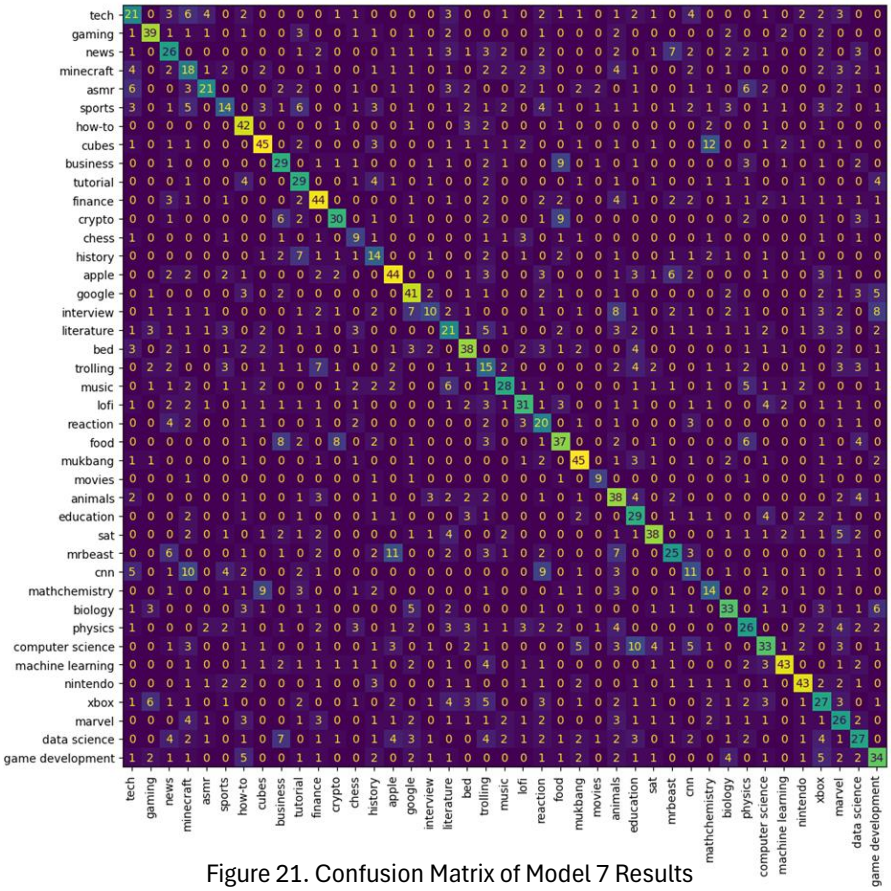


Figure 21. Confusion Matrix of Model 7 Results

Training Accuracy: 61.83% Validation Accuracy: 45.75% Testing Accuracy: **44.97%**

This model resulted with test accuracy of 44.97%, which may appear low. However, the presence of large number of classes and the above obtained confusion matrix explains the low accuracy and it shows that the results can be considered as satisfactory.

MODEL 8 (google-use) (with bidirectional LSTM)

In this model Google's USE was used with bidirectional LSTMs. Adam optimizer with learning rate 0.001 was used. As loss function, categorical cross entropy loss was used. The model was trained for 30 epochs, with batch size 32.

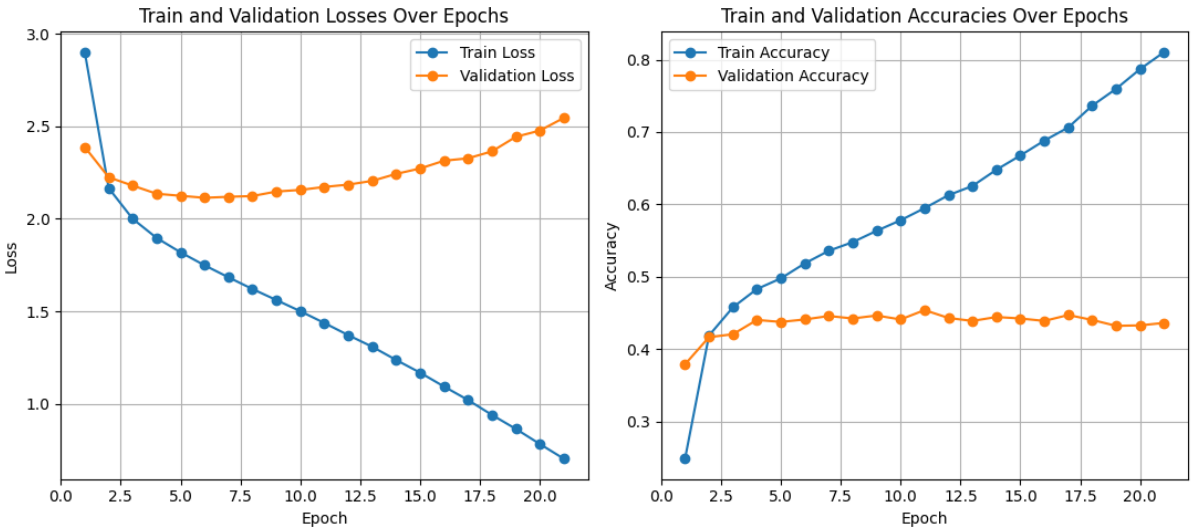


Figure 22. Train & Validation Losses and Accuracies Over Epochs

Early stopping mechanism based on validation accuracy was used with patience of 10 epochs. In this case, the weights at the 11th epoch are used.

Training Accuracy: 59.49% Validation Accuracy: 45.41% Testing Accuracy: **46.28%**

Similarly, this model resulted with test accuracy of 46.28%, which is not satisfactory. The presence of large number of classes and the above obtained confusion matrix explains the low accuracy and it shows that the results can be considered as satisfactory. Also, difference between training accuracy and test accuracy indicates that there is an overfitting. This model provided slightly better results than the previous model.

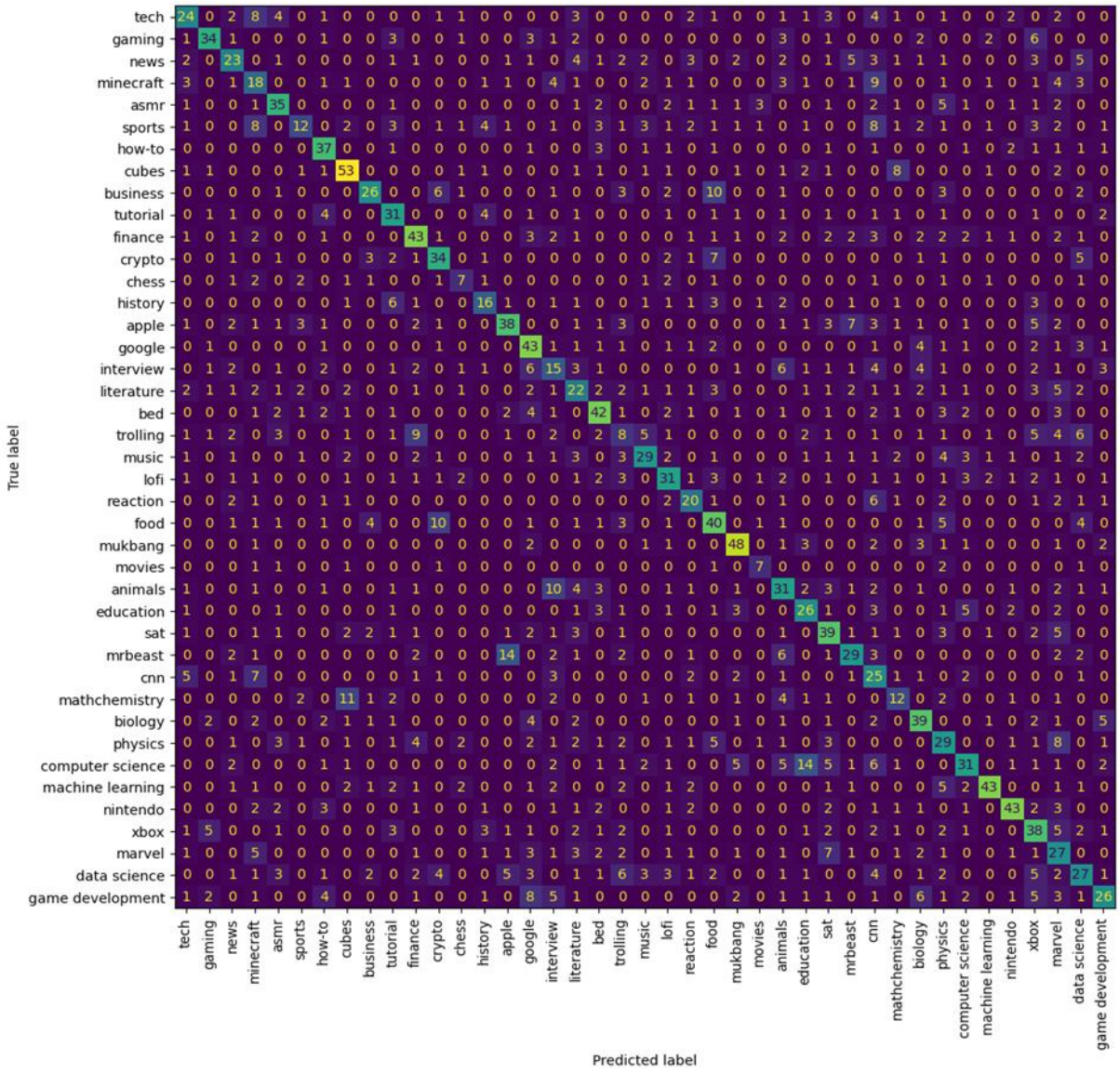


Figure 23. Confusion Matrix of Model 8 Results

5. Analysis and Comparison of Models

5.1. Sentiment Analysis Models

Sentiment Analysis Models are those which are majorly designed for classification purposes on the text data, namely comments, to classify them as positive, negative, or neutral. So, in this project, many models and techniques have been experimented with, which are coming up with their advantages and limitations. The major models are developed using BERT, Naive-Bayes and other Artificial Neural Network models.

BERT (Bidirectional Encoder Representations from Transformers)
For this project, BERT was fine-tuned for sentiment classification. However, the performance of this model is not up to the mark. The model resulted with testing accuracy of 62%. The main problem is that the model classifies all comments as a positive comment, leading to a possible bias in the training data or lack of handling the complexity of the dataset by the model.

Naive-Bayes with TF-IDF
Using the TF-IDF vectorization with the Naive-Bayes model, the test accuracy was 66.63%. After hyperparameter tuning, accuracy reached 72.07%, which tells the notion that Naïve Bayes works in this kind of sentiment analysis because of its simplicity and efficiency.

Artificial Neural Networks (ANNs)
Several ANN models were tested using different word embeddings:

Model 1 (Glove-Wiki-Gigaword-50): The model achieved a testing accuracy of 71.80% after early stopping on the 16th epoch.

Model 2 (Word2Vec-Google-News-300): This model gave an increased result showing a testing accuracy of 75.24% – hence showing the word2vec embeddings' potential in capturing the semantic relationships present in the text.

Model 3 (Custom Embedding Layer): This model used a custom embedding layer with the LSTMs, and its testing accuracy was 63.22%.

Model 4 (Glove-Wiki-Gigaword-50) (with LSTM): This model has same embedding method with model 1, but different architecture with LSTM layers in it. It has reached 63.53% testing accuracy.

Model 5 (Universal Sentence Encoder - USE): This model achieved a notable accuracy of 81.31%, the highest among all tested models. The embeddings of USE could capture the semantic representation of sentences, fitting for such a sentiment analysis application.

Model 6 (USE with Bidirectional LSTM): Adding bidirectional LSTMs to the USE model, capturing context from both directions, led to improved understanding and classification of sentiment. This model, too, attained similar high accuracy of 80.15%.

The comparison of these models underlines the importance of choosing the right embeddings and architectures. While BERT failed due to high computational needs and the possibility of overfitting, Naive-Bayes offered a balance of simplicity and efficiency. ANN models with pre-trained embeddings, especially USE, showed supreme performance, emphasizing the value of contextual embeddings in sentiment analysis.

5.2. Keyword Prediction Models

The task of keyword prediction models is to predict the keywords relevant for the YouTube videos based on the comment content. The large number of possible classes for keywords makes this a very challenging task. The main models were those with Google's Universal Sentence Encoder (USE) and its corresponding versions with dense layers and Bidirectional LSTMs.

Model 7 (USE with Dense Layers)**:** This model utilized Google's USE with dense layers. The testing accuracy was 44.97%. Though it seems low, the performance of the model can be considered satisfactory, considering the complexity and the large number of classes (41 different keyword classes).

Model 8 (USE with Bidirectional LSTM)**:** The combination of the USE embeddings with Bidirectional LSTMs allows one to capture contextual information from both directions. The obtained test accuracy was slightly better at 46.28%.

The complexity of the keyword prediction task can be inferred from the relatively low accuracies in this case. The USE embeddings were a strong base, with a lot of semantic meaning in the sentences. But both the dense layer model, Model 7, and the Bidirectional LSTM model, Model 8, were troubled by the large number of classes. Model 8, the Bidirectional LSTM model, had a slight improvement, implying that, with the bidirectional context, the understanding of relationships between words and hence the prediction of keywords improves. The challenges lie in overfitting, which means there would be a need for more sophisticated regularization techniques or augmentation of the dataset to be done.

6. Conclusion

After conducting numerous experiments with different embeddings, models, algorithms, and approaches, acceptable results were achieved in sentiment analysis and keyword prediction. The best results were achieved with models using Google's Universal Sentence Encoder, which yielded an accuracy of 81%. Following this, the same approach was applied to keyword prediction. It is believed that these results, although worse than the metrics of the previous task, are influenced by the fact that in the dataset for keyword prediction, there are 41 distinct classes of keywords. The confusion matrices for these models support this opinion, as they indicate the model's ability to handle complex classification tasks despite the overall lower performance metrics.

There were several difficulties due to computation and time limits, which eventually led to the main goal not being reached: generating comments from comments. This proved to be especially difficult in the context of this project. The option of using GPT-2 for comment generation did not work, and the attempt to use BERT for training failed due to its high demands in terms of time, memory, and computational resources.

In the future, the developed AI model will be deployed by creating a user-friendly interface for content creators, who will then be able to use the comment generation and sentiment analysis features. In addition, the model could be integrated directly into the YouTube comment section for real-time insights and suggestions. The project will collect feedback and improve the model over time with real-world usage patterns. To ensure continuous improvement, the project will collect feedback from users and develop the model according to the collected feedback on real-world usage patterns. Further, the application could be extended not only for YouTube but also for other social media platforms, having the model adapted and supporting multiple languages to reach a wider audience.

7. References

[1] Mehta, T., & Deshmukh, G. (2022). YouTube Ad View Sentiment Analysis using Deep Learning and Machine Learning. International Journal of Computer Applications.
[2] https://www.kaggle.com/datasets/advaypatil/youtube-statistics [Accessed: 06.06.2024]
[3] Gunasekaran, K. P. (2023). Exploring Sentiment Analysis Techniques in Natural Language Processing: A Comprehensive Review.
[4] Zhang, Y., Jin, R., Zhou, Z.-H. (2020). Understanding Machine Learning Techniques for Sentiment Analysis