Spring 2021

Clickbait Detection in YouTube Videos

Ruchira Gothankar
San Jose State University

Clickbait Detection in YouTube Videos

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ruchira Gothankar

May 2021

The Designated Project Committee Approves the Project Titled

Clickbait Detection in YouTube Videos

by

Ruchira Gothankar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2021

Mark Stamp	Department of Computer Science
Fabio Di Troia	Department of Computer Science
Mike Wu	Department of Computer Science

ABSTRACT

Clickbait Detection in YouTube Videos

by Ruchira Gothankar

YouTube videos often include captivating descriptions and intriguing thumbnails designed to increase the number of views, and thereby increase the revenue for the person who posted the video. This creates an incentive for people to post clickbait videos, in which the content might deviate significantly from the title, description, or thumbnail. In effect, users are tricked into clicking on clickbait videos. In this research, we consider the challenging problem of detecting clickbait YouTube videos. We experiment with logistic regression, random forests, and multilayer perceptrons, based on a variety of textual features. We obtain a maximum accuracy in excess of 94%.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my project advisor Dr. Mark Stamp for guiding me and providing me suggestions and comments throughout the course of the project. He has helped me by providing constant feedback without which the project's successful completion was not possible. I would also like to thank my committee members Prof. Fabio Di Troia and Prof. Mike Wu for their time and guidance.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

Introduction

Today, web content is increasingly popular and people rely on information obtained from the internet. With the diversity of the sources available, the amount of time spent on the internet has increased. Many platforms provide a medium where virtually anyone can publish information that is accessible to a large number of people. The credibility of such information is a relevant consideration.

Online sources of information include blogs, videos and social media, among others. Many applications have been developed to personalize information. Along with sharing information, these applications also act as a medium to generate revenue. However, unscrupulous people can use false information to increase their viewership and gain attention of users which leads to trust issues. Clickbait is a false and deceptive information which lures a user to click a link, watch or read the information. It aims to exploit a user's curiosity by providing misleading though captivating information [1]. Clickbait has become a marketing tool bsy many people, to entice users and thereby generate revenue for themselves. Publishing eye-catching information to manipulate and trick users is a common practice to increase the viewership and spread brand awareness. A clickbait can be an image, a sensational headline or a misleading video/audio content. While clickbait sources help in gaining attention, there are a lot of disadvantages and negative ramifications to it. This not only wastes the time of the viewers but also affects the trustworthiness [2].

YouTube is one such video publishing platform where users upload videos and share them. When uploading a video, a user adds a title, a description, and a thumbnail. Users can view the title and thumbnail of the video before clicking the video. These become crucial parameters on which users base their decision as to whether to click a video or not. Many YouTubers use clickbait title and thumbnails

that might deviate from the actual content to increase viewership for a video, and thereby generate more revenue.

A recent example includes the COVID-19 pandemic, where individuals have posted misleading health-related content, including some fake cures for COVID-19. Some other common examples of clickbait titles phrases are "You'll Never Believe What Happened Next...", "The 10 documentaries you should watch before you die", "You Can Now Travel Abroad Without Having to...","You Won't Believe..." [3]. Figure 1 shows an example of clickbait video on YouTube.

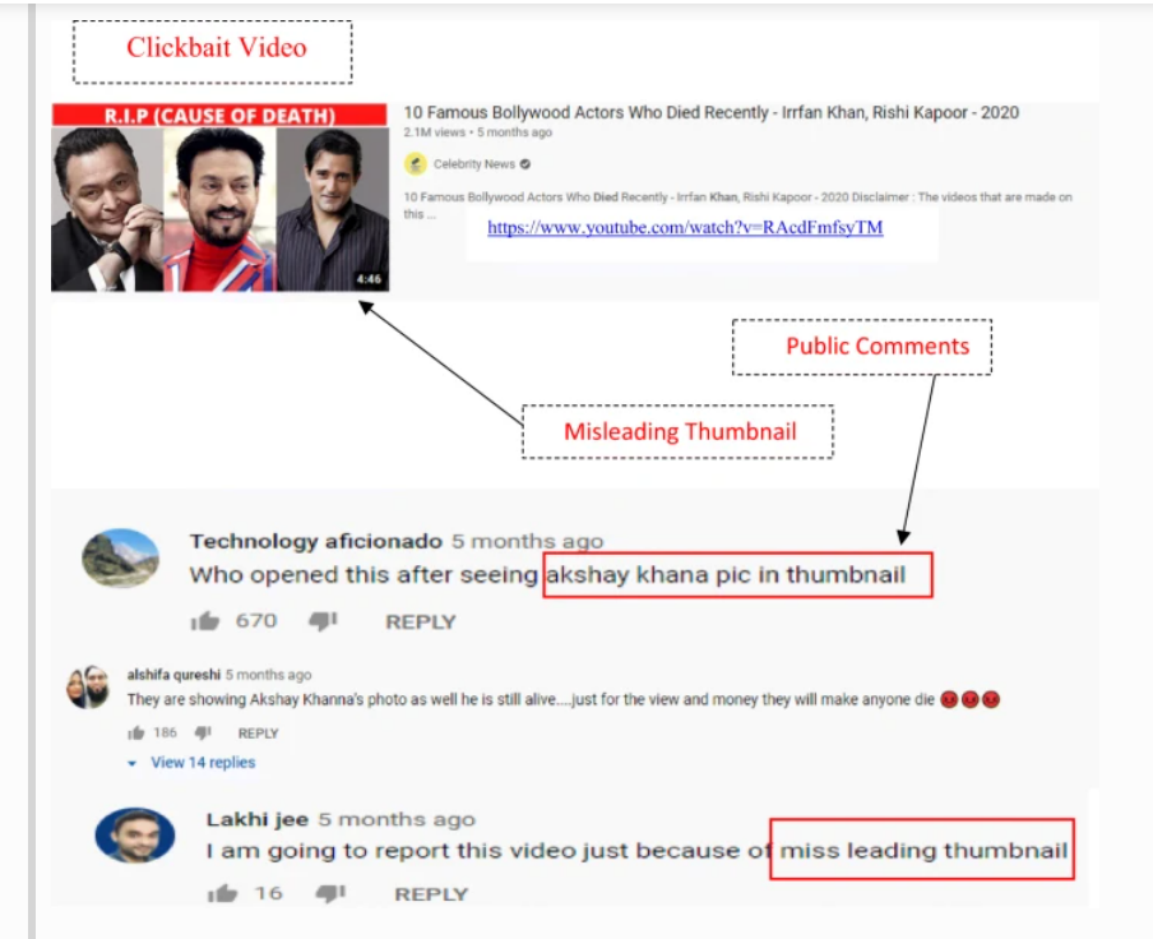


Figure 1: Example of Clickbait Video on YouTube [4]

The clickbait problem is somewhat similar to that of spam and fake news detection.

Spam, which is unsolicited bulk email, often includes misleading messages that are sent to deceive users by redirecting them to websites for the purpose of advertising or attacks. Considerable research has been focused on detecting spam and fake news. In this research, we are concerned with detecting clickbait YouTube videos. Previous work suggests that many YouTubers rely on clickbait to increase their revenue. The YouTube platform relies on users to manually flag suspected malicious or clickbait content. A more automated approach would clearly be desirable. In this paper, we consider machine learning and deep learning based solutions to the clickbait detection problem.

The remainder of this report is organized as follows. Chapter 2 considers relevant previous work and background topics related to natural language processing (NLP). In Chapter 3, we discuss our experimental setup, including the datasets used. Chapter 4 contains our experimental results, and we provide analysis of these results. In Chapter 5 we give our conclusions and we discuss possible directions for future work.

CHAPTER 2

Background

This section discusses about the relevant works done so far in this field. It mainly focuses on clickbait detection, fake news detection, image forgery detection and hoax detection. Apart from that, this chapter also discusses the advancements in NLP over the years.

2.1 Related Work

This section discusses and analyses the performance of previous works done so far on clickbait, fake news, forgery and hoax detection. Misleading content is present in the internet in multiple forms and is often used in different contexts interchangeably. For example, misrepresenting facts like false celebrity death stories are a common form of hoaxes [4], while spreading false information through images is a form of forgery. The clickbait, is another form where a viewer is lured to know more about the information.

2.1.1 Clickbait detection

Some of the related works in clickbait detection are described below. Chakraborty et al. in 2016 [5], implemented an ML classifier to detect clickbait. They also implemented a browser extension to help readers. They used the headlines from the Wiki-news corpus and used 18,513 articles as legitimate posts. For the clickbait posts they used articles from popular domains which posted illegitimate posts. To train their classifier, they used a set of fourteen features spanning linguistic analysis, word patterns, and N-gram. They achieved an accuracy of around 89% using Support Vector Machine (SVM) classifier to detect and avoid clickbait.

In 2017, Elyashar et al. [6], discussed an approach focused on feature engineering. Their work focused on detecting clickbait posts in online social media. They performed linguistic analysis using a machine learning classifier which could differentiate between

legitimate and illegitimate posts. The dataset used for analysis was provided by Clickbait Challenge 2017. The results of their experiments suggested that the malicious content is longer than the benign content. They also concluded that the title of a post played an important role to classify a clickbait.

Glenski et al., in 2017 [7], developed a network model which is a linguistically infused network to detect fake tweets. This model which was based on Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) used the text of tweets, images and description for training. Pretrained embedding model GloVe was used as the embedding layer. They achieved an accuracy of 82%. Zhou in 2017 [8], proposed a self attentive neural network model using Gated Recurrent Units (GRU) for predicting fake tweets. They performed multi classification using the annotation scheme. They ranked first in the Clickbait Challenge 2017 with an F1 score of 0.683.

2.1.2 Fake news detection

Fake news detection is a type of misinformation detection which has received huge amount of attention over the years. In this, the text content in a news is analyzed to check if the statement is true or not. In [9], Ahmad et al. implemented an ensemble model which involved a combination of multiple machine learning algorithms like Random Forest, Multi layer perceptron, Support Vector Machine (SVM) to detect fake news and achieving an accuracy of 92%. The worked on the linguistic features. They used XGBoost as the ensemble learner which improved the performance.

Thota et al. [10], presented a paper on detecting fake news using natural language processing. The used TF-IDF and Word2Vec with a dense neural network on the news headline to classify if a news is fake. In another paper on fake news detection, Jwa et al. [11] have implemented a model using Bidirectional Encoder Representations from Transformers (BERT). The deep contextualizing nature of BERT has given the

best results. It was useful in determining the relationship between the news headline and the body of the news.

2.1.3 Forgery Detection in Images and Videos

Image forgery detection is another type of misinformation detection where malicious information is conveyed through images. In 2018, Zhang et al. [12], developed a tool which was a fauxtography detector which could detect and track down images which were misleading on the social media. Many such approaches could detect malicious content however they were not suitable for videos.

Palod et al. [13] passed the pretrained Word2Vec comment embeddings through an LSTM network. A fakeness vector was generated using 30 such phrases and was trained on LSTM. They achieved an F-score of 0.82. In 2019, Shang et al. [2], proposed a model which involved network feature extraction, metadata feature extraction and linguistic feature extraction to detect clickbait in YouTube videos. The network feature extraction used comments in the videos and extracted semantic features. In the linguistic feature extraction they used the document embedding for comments using Doc2Vec. The metadata module focused on the metadata for the video. In 2019, Reddy et al. [14] implemented a model using word embedding and trained on Support Vector Machine (SVM). In [15] Dong et al., have proposed a deep similarity-aware attentive model which focuses on the relation between the titles that are misleading and the target content. This method was quite different from the traditional feature engineering and seemed to work well. Setlur in [16] worked on a semi-supervised confidence network along with an gated attention based network. Even with a small labeled dataset, this method gave good results.

In many of the above approaches, only the textual information like title and description along with the metadata features have been taken into consideration while

training a model. However, in [2], they have also used comments to extract features. Also the embedding layers of Word2Vec, BERT and Doc2Vec have been commonly used across all the mentioned implementations.

In this project, we intend to experiment with multiple embedding layers like BERT, DistilBERT, Word2Vec. BERT has proven to be efficient because of its deep contextualizing nature [11]. Also, a combination of multiple models basically known as ensemble learning has given good results in [10]. Ensemble models like Random Forest classifiers include a combination of results from multiple decision trees is used to find a better solution. Semi-supervised learning is best suitable in this scenario where there is small labeled dataset. Also, as mentioned in [16], this approach gave good results.

2.1.4 Hoax detection

The new reports in which the facts are misrepresented are known as Hoax. These reports provide deceptive information to readers and present it as legitimate facts. One such example of hoaxes include a fake story about a celebrity death. In [17], the authors have proposed a technique which uses logistic regression for classifying hoaxes. In the model proposed, they have used features based on user interaction and have achieved an accuracy of 99%. Whereas, in [18] Zaman et al. employed a Naïve Bayes algorithm which uses the feedback from users as an input to verify a news as hoax or true. Kumar et al. [19], have proposed an method which uses using Random forest classifier to classify credibility of the articles on Wikipedia. They achieved an accuracy of 92%. Hoax detection is an area which needs more research and is less explored.

2.2 Advancements in NLP

Natural Language Processing (NLP) dates back to 1950 when Alan Turing published an article "Computing Machinery and Intelligence", in which he developed a Turing Test in order to determine a machine's ability to demonstrate intellectual behaviors close to those of a person [20]. NLP is the ability of a machine to process and understand the language of a human. It is used to solve many real world problems like machine translation, question answering, predicting words. Figure 2 shows a timeline of advancements in NLP. In the early 1990's, statistical and probabilistic

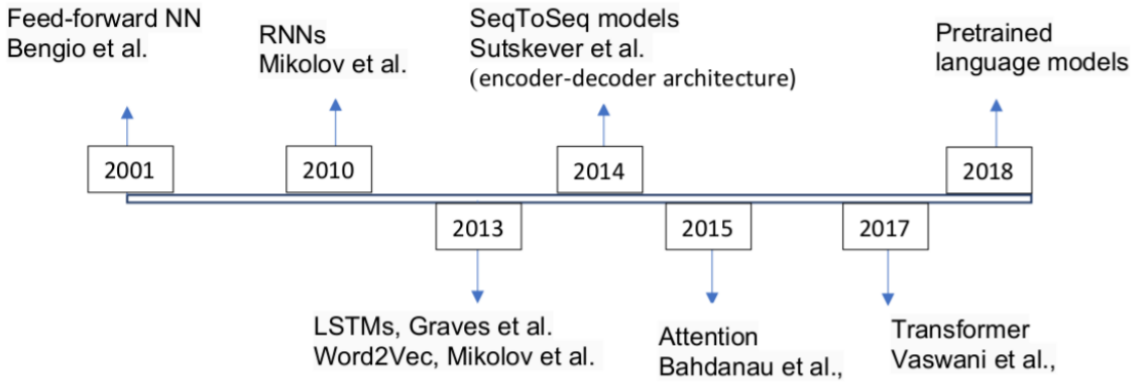


Figure 2: Advancements in NLP Over The Years [21]

approaches were employed to train algorithms. But with the invent of Web, the amount of data grew. In 2001, Bengio et al. proposed the first Feed Forward Neural Network. Later on, Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) were introduced [22]. In early 2012, techniques like Latent Semantic Indexing (LSI), Latent Semantic Analysis (LSA), Support Vector Machine (SVM) were quite popular which introduced the concept of semantic similarity. Part of Speech (POS) tagging was the common approach used to tag words in the sentences.

In 2013, Tomas et. al. introduced Word2Vec word embedding which is used

for language modeling [23]. This became a popular research area by 2016. Word embeddings using Word2Vec can be constructed in two ways: Common Bag of Words (CBOW) and Skip Gram as shown in Figure 3. Both of these approaches use neural network. Word2Vec was trained on 1.6 B dataset of words. The cosine distance between words was considered as an important parameter [24].

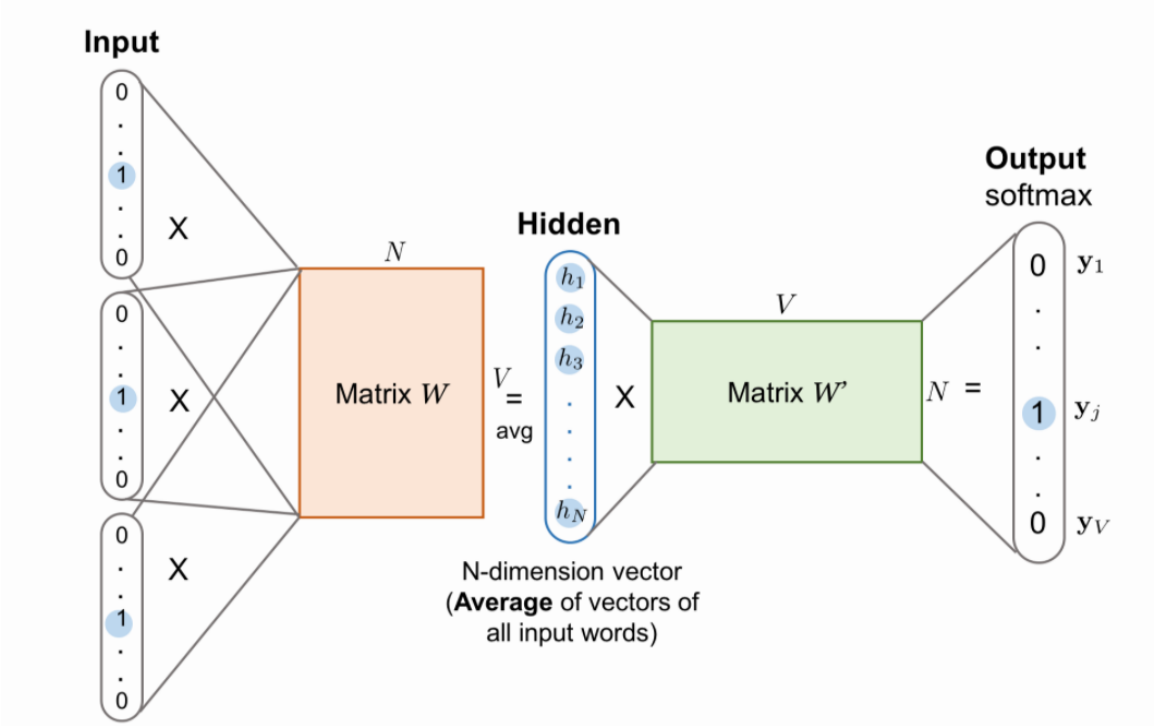


Figure 3: Word2Vec CBOW Model. The input and the output vectors are one-hot encoded word representations. The hidden layer represents the embeddings [25]
.

Global Vector for Word Representation (GloVe), was introduced in 2014 and it was an attempt to combine the benefits of LSA, LSI and Word2Vec. It is based on occurrence of a word in the entire corpus. With the onset of deep learning, in 2014, CNNs and LSTMs became popular for NLP related tasks [22]. LSTM is a type of Recurrent Neural Network (RNN) which works on data that is sequential. Gated Recurrent Unit (GRU) is another variant of LSTM introduced in 2014 which is lighter

and has fewer parameters. Models based on RNN takes textual data as input in a sequential manner.

Sutskever et al. [26] proposed the sequence to sequence learning which used encoder-decoder architecture based on neural network, became another milestone in NLP. Encoder-Decoder is the main language modeling framework in today's NLP tasks. The main challenge is using RNNs in learning long term dependencies. Also, lack of parallel computation is a disadvantage in training large datasets [21]. Attention, a concept in deep learning was proposed by Bahdanau et al. [27] in 2015 to overcome the limitation of fixed vector length for input sentences in sequence to sequence model [28]. The weights in attention provides information about the importance of a part of sentence while making a decision.

To overcome the complexity of attention, Transformers were introduced in 2015 by Google [29]. Transformer includes multiple stacks of encoder-decoder architecture and at each step in the processing, the model takes the output of the previous step as an input. Figure 4 shows the architecture of a transformer where the decoder is on the right and the encoder is on the left. Initially, the input tokens are embedded into the embedding vectors. Since this model does not have any RNN units, in order to store the context and the position of the word in the sentence, these position indices are stored in the n-dimensional vector space in the form of embeddings. The positional encoding uses sin-cosine functions for each word. There are 3 fully connected layers in attention mechanism. It involves inputs key K, value V and query Q which is a matrix of queries and defines weights for words based on all the words in K, a vector representation for all words to multi-head attention [29]. The other processes includes context fragmentation, multiple parallel attention layers. Some example of deep learning models using transformers include BERT, RoBERTa, mBERT and DistilBERT.

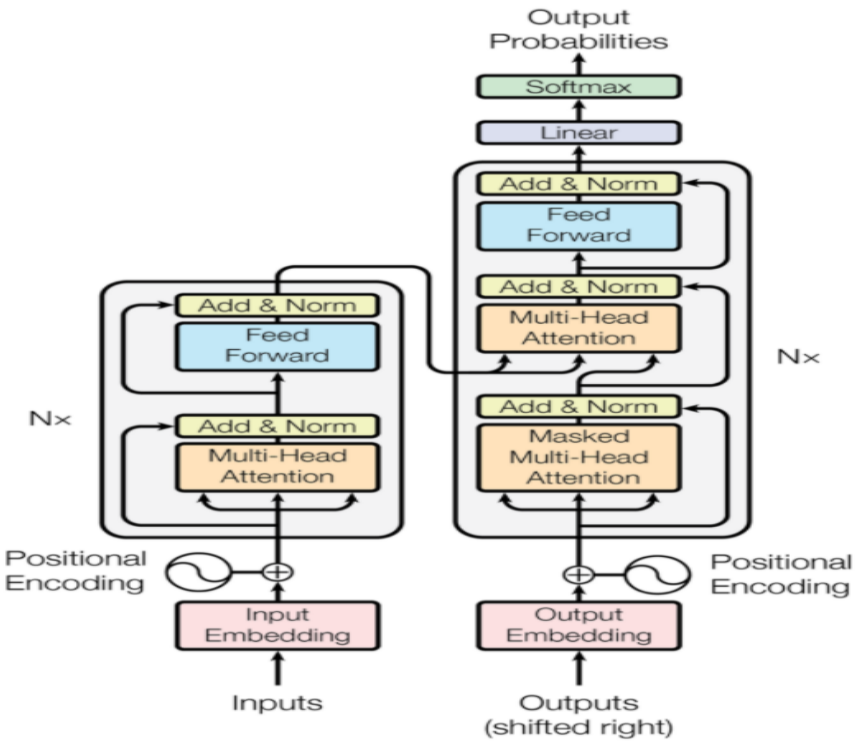


Figure 4: Architecture of a Transformer [29]

Bidirectional Encoder Representations from Transformer (BERT) uses transformer which is based on attention mechanism to learn the contextual relation between words. It involves an encoder which reads the input and decoder which predicts the output. It is called bidirectional because instead of reading input sequentially from a specific direction, the transformer encoder in BERT reads the entire sequence of words altogether. This helps in learning the context of the word based on the words present on its left and right. BERT uses an approach called Masked LM (MLM). In this approach it masks words randomly and tries to predict them based on the words present in its surrounding. The textual input in BERT requires a specific format. All the sentences needs to be formatted by appending special tokens to it. A [CLS] and a [SEP] token also called as token embeddings, are added to the start and the end of the sentence. Token embeddings are used to identity the beginning and end of a

sentence. Segment embedding which is a marker to differentiate between sentences is appended between all the sentences. Positional embedding for each token which indicates the position of the token is added. Figure 5 shows the input pattern used for the BERT model.

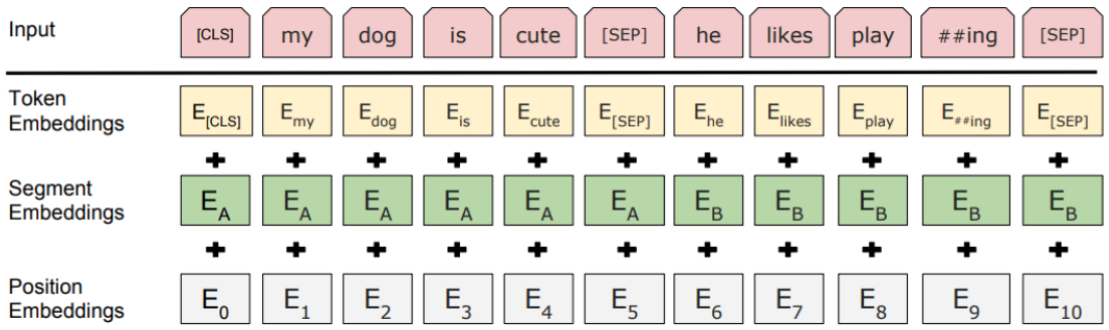


Figure 5: Input for the BERT Model [30]

.

BERT has four pretrained versions with different layers, hidden nodes and parameters. BERT model can be fine tuned for a specific task by adding additional layers. DistilBERT is a lighter and a faster variant of BERT Model.

2.3 Techniques

In this section, we discuss multiple techniques used in this project. We have performed experiments on techniques like Logistic regression, Random forest classifier, Multi layer perceptron.

2.3.1 Logistic regression

Logistic regression is one of the basic methods for classification in machine learning. It is a type of linear classifier. It is a supervised learning algorithm which is used for categorical data where some parameters which depend upon the input features and the output is a categorical prediction. In logistic regression, a sigmoid function is fitted on the data and it gives a S-shaped curve with value ranging from $0 < p < 1$.

The formula for the sigmoid function is shown in the equation

$$\sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{1}$$

Logistic regression requires a dependent variable, a mean function to generate predictions and a link function which can transform the mean function back to the distribution of the dependent variable. Logistic regression assumes that there is no correlation between the independent variables. It uses a logit function called as log-odds. The ratio of odd is the probability of odds of a event S with T and odds of event S without T. The clickbait detection problem is a type of binomial logistic regression where output can be either zero or one [31].

2.3.2 Random forest classifier

Random forest classifier is based on decision trees. It involves a large group of decision trees which operate together like an ensemble. Each tree in this group gives a prediction for a class, and the class which gets the most number of vote counts is the output. Figure 6 represents the working of random forests. Decision trees can lead to overfitting, but random forests prevent overfitting by selecting random subset of features.

The important hyper parameter in random forest are n-estimators, n-jobs, max-features, min-sample-leaf. n-estimators represent the count of trees that the algorithm builds before taking a vote. Usually, adding more trees increases performance at the cost of computation time. max-features is the count of features that the algorithms takes in to split a specific node. n-jobs is the total count of processors that work in parallel. Random forest uses ensemble technique called Bagging to make a prediction. The data for each tree is randomly sampled with replacements and use different features to make a decision [32].

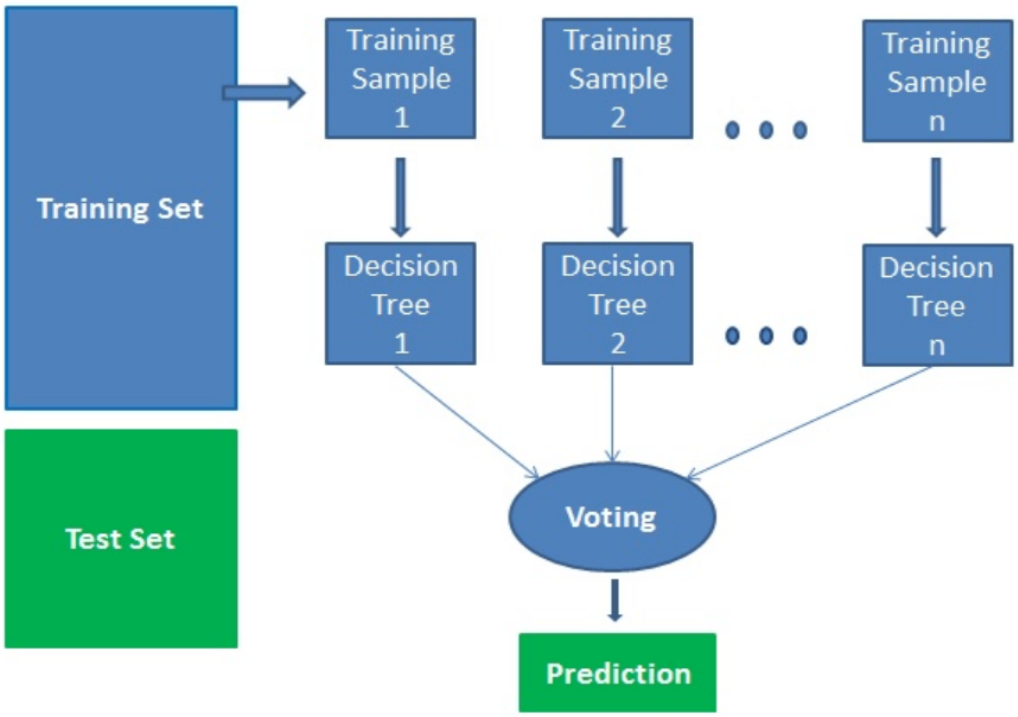


Figure 6: Random Forest Algorithm

2.3.3 Multi layer perceptron

A Multi layer perceptron is a type of feed forward neural network which includes input, output and hidden layers. The output layer does the task for prediction or classification. A single neuron model is called as a perceptron. Figure 7 shows a simple Multi layer perceptron model. This model involves three processes - forward pass, loss calculation and backward pass. In forward pass, the input is passed and multiplied with the weights and a bias is added at every layer. In the next step the loss is calculated based on the predicted output and the actual output. Lastly, in the backward pass, the loss is back-propagated to update the weights of the model. The common techniques applied to a Multi layer perceptron model includes Regularization, Activation, Optimization [33].

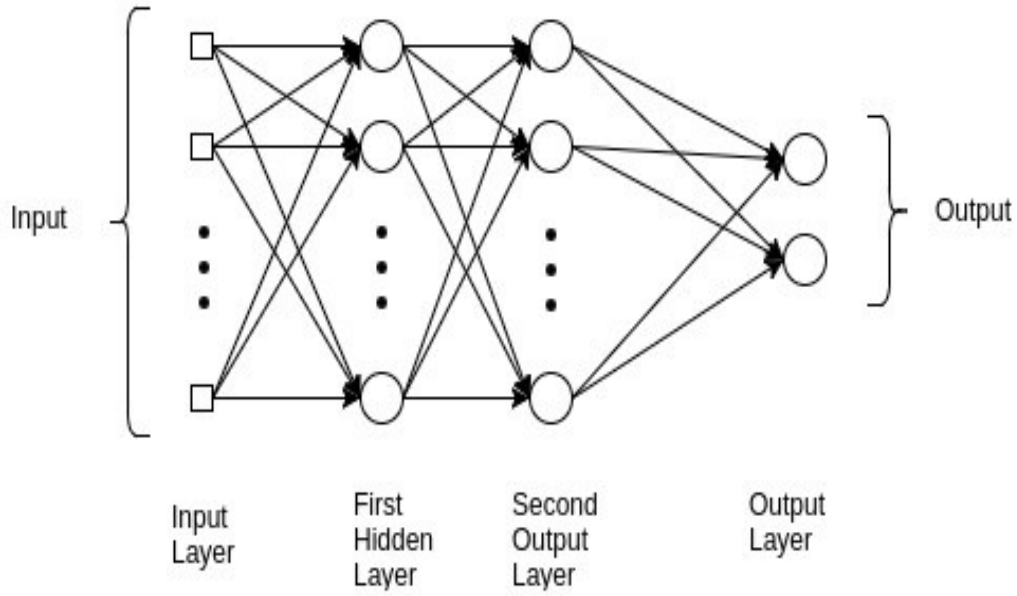


Figure 7: Multi Layer Perceptron Model

2.3.3.1 Regularization

Neural network models are prone to overfitting. The have the tendency to remember the train data and therefore they fail in predicting the test data correctly. Therefore, penalties are added in a network to reduce overfitting. Dropout is a type of regularization where some number of outputs from the layer are randomly dropped and ignored [33].

2.3.3.2 Activation

Activation function also called as transfer function is used to determine the output of layer in the neural network. There are multiple types of activation functions like Tanh, Sigmoid, ReLU, Leaky ReLU [33]. In this project we have experimented with activation functions like ReLU, Tanh.

2.3.3.3 Optimization

Optimizers are used to reduce the loss in a neural network. In this technique, multiple attributes of the network like weight, learning rate is modified to improve the network. Multiple techniques like gradient descent, stochastic gradient descent, adaptive gradient, adam are used [33].

CHAPTER 3

Implementation

This chapter includes all the details of implementation of the project. It discusses the setup used to train and execute the machine learning models, approaches for the experiments and techniques used for training the model.

3.1 Setup

In this project, we have used multiple conda virtual environments for each implementation. Conda is an open source package and environment management system which runs on multiple OS [34]. The following are the configurations for the host machine:

- Model : ASUS ZenBook
- Processor : Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
- RAM : 16.0 GB
- System Type : 64-bit OS
- Operating System : Windows 10

All the training and experiments are done on the host machine.

3.2 Approach

The clickbait detection task takes a set videos as input for detecting videos which do not represent the event it is supposed to. This problem is formulated as a binary classification problem where for each video the algorithm determines if the video is clickbait or non-clickbait. The information from multiple modalities like title, description, statistics, comments is combined and fed to the classification model. The performance is then evaluated and analyzed by multiple measures like F1 Score, Recall, Precision. There are three main modalities considered in this project. The first component involves extracting features from the profile of the user (subscription count, view count, video count). The second feature is based on extracting textual

features from the video (title, description). The third component involves extracting statistical features related to the video (likes count, dislike count, like-dislike ratio, view count, comment count). Lastly the classification model performs the binary classification(non-clickbait/clickbait) based on the features extracted. The overview is shown in Figure 8.

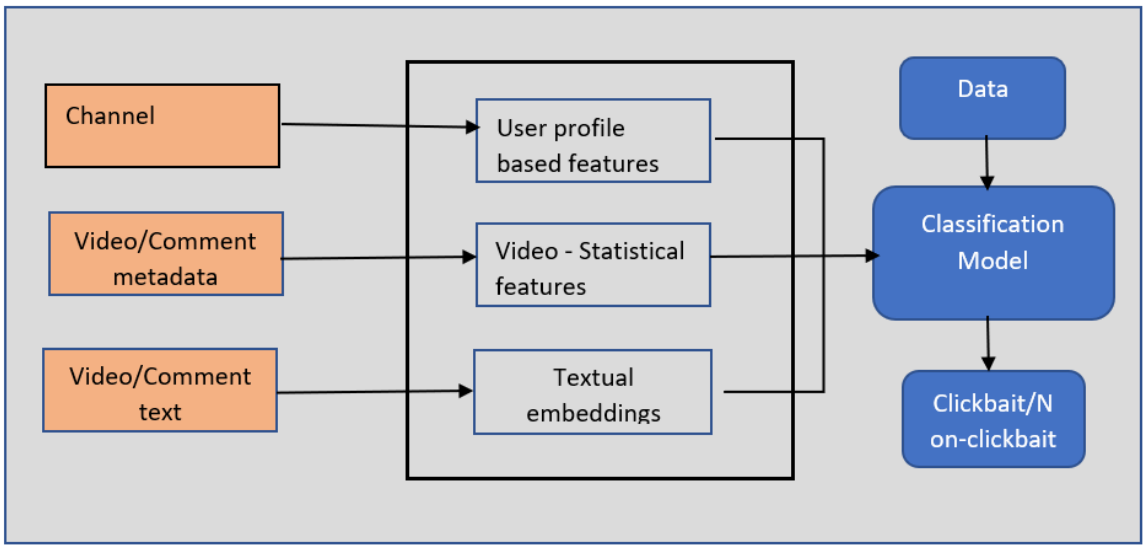


Figure 8: Overview of the clickbait classification model

3.2.1 User profile or YouTube channel features

Features like number of subscribers for a YouTube channel, number of likes or upvotes for a channel, age of channel provides a lot of information regarding the reputation of the channel and the videos. These statistical features represent the response of the viewers for the channel. From the previous related works, it has been seen that videos which are clickbait has a smaller number of subscribers and likes [4].

3.2.2 Video/Comment statistical features

Usually the number of views for the clickbait and non-clickbait videos are quite similar [35]. This is why the viewers are quite frustrated after they watch the video. Features like dislike ratio, favorite count, video age, view count, comment count

adds useful information in determining the credibility of a video. In clickbait videos, sometimes the uploader disables the comment section. This provides informative clues about the video [4].

3.2.3 Textual Features

Textual features involve using Headline of the video, description of the video, comments by the viewers for the video as the input. YouTubers who upload clickbait usually employ techniques which are deceptive. They use catchy and exaggerated phrases for title and description of the video. The common phrases are "viral", "top", "won't believe", "epic". These texts are tokenized and embedded using multiple embedding techniques like Word2Vec, BERT, DistilBERT in the classification model.

3.3 Dataset

Every month, billions of people visit YouTube and the video are watch for over a billion hours. A lot of videos are also uploaded. YouTube is a platform where people can generate revenue by uploading videos and gaining viewership for their videos.

YouTube Data API provided by Google allows users to crawl through the videos and extract necessary information. The search query helps in searching for videos matching a specific criteria. There are multiple API's for each tasks. Hence, no single API could retrieve all the information needed in one go. We developed a small application to fetch all the required data in the format needed. In this project, the evaluation is done on a dataset of 8219 videos of which 4300 are non-clickbait and 3919 are clickbait videos. The dataset has been crawled from the Google YouTube API for the list of video IDs fetched from a Github source [36]. These sources were randomly picked and manually verified. The statistics for the parameters are shown in Table 1.

Table 1: Dataset statistics

Data Item	Distribution		
	Min	Mean	Max
Title length	10	54	107
Description length	15	1131	5162
View count	21	5,660,978	2,543,466,463
Comment count	0	522	49,060
Like count	0	49,615	13,542,232
Subscriber count	977	10,200	23,695,417
Dislike count	0	1320	516,171

3.4 Techniques

In this project, we have experimented multiple techniques including multiple language modeling techniques. We have used Word2Vec, BERT and DistilBERT for word embeddings. Architecture for the individual models is also shown. Technique like grid search has been used for training and building model using the best set of parameters.

3.4.1 Experiment I - Logistic Regression with Word2Vec

In this experiment we used a Word2Vec model by gensim to the the vector representations for words in the dataset. A logistic regression model is trained on these embeddings along with the additional features like comment count, like count, dislike count, subscription count for the channel.

3.4.2 Experiment II - Random Forest Classifier with Word2Vec

In this experiment a Random forest classifier is trained on the Word2Vec embeddings for the dataset. We used the Word2Vec model provided by Gensim. The model has been tested on multiple set of inputs. We have used a grid search to search for best set of hyper parameters. The values for n-estimators is 10, 20, 30, 50, 100. The first set of input features are title, description, and meta data features like comment

count, like count, dislike count, subscription count for the channel.

3.4.3 Experiment III - Simple Multilayer Perceptron with Word2Vec Embedding

We have used the Word2Vec model implemented by Gensim. This experiment involved creating word tokens, followed by removing punctuations and stop words. The Gensim Word2Vec API is initialized with a window size of four and list of input sentences. The embedding for title and description is concatenated with the meta data features of the video and is fed to a neural network. We have experimented with a simple Multi layer perceptron and Word2Vec Embedding. The batch size is 10 for 40 epochs. The activation functions used are ReLU and Sigmoid. Figure 9 represents the plot for the model. It shows the overall architecture for the model. Two input embedding layers for textual data like title and description are then concatenated together. The output from the dense layer is then flattened and concatenated with the input for the meta data features. After this, a fully connected layer is used for classification.

3.4.4 Experiment IV - Multi layer perceptron with DropOut and Batch Normalisation and Word2Vec Embedding

This experiment is an optimization of the previous experiment. In this model, additional dense layers along with batch normalization and Dropout of 0.5 is added. We have used Parametric Rectified Linear Unit, or PReLU, as an activation function. The batch size is 10 for 40 epochs. Figure 10 represents the plot for the model. In this model, the output from the embedding layers for the textual data is concatenated, followed by a fully connected dense layer and batch normalization and activation. This output is then flattened and concatenated with the meta data features.

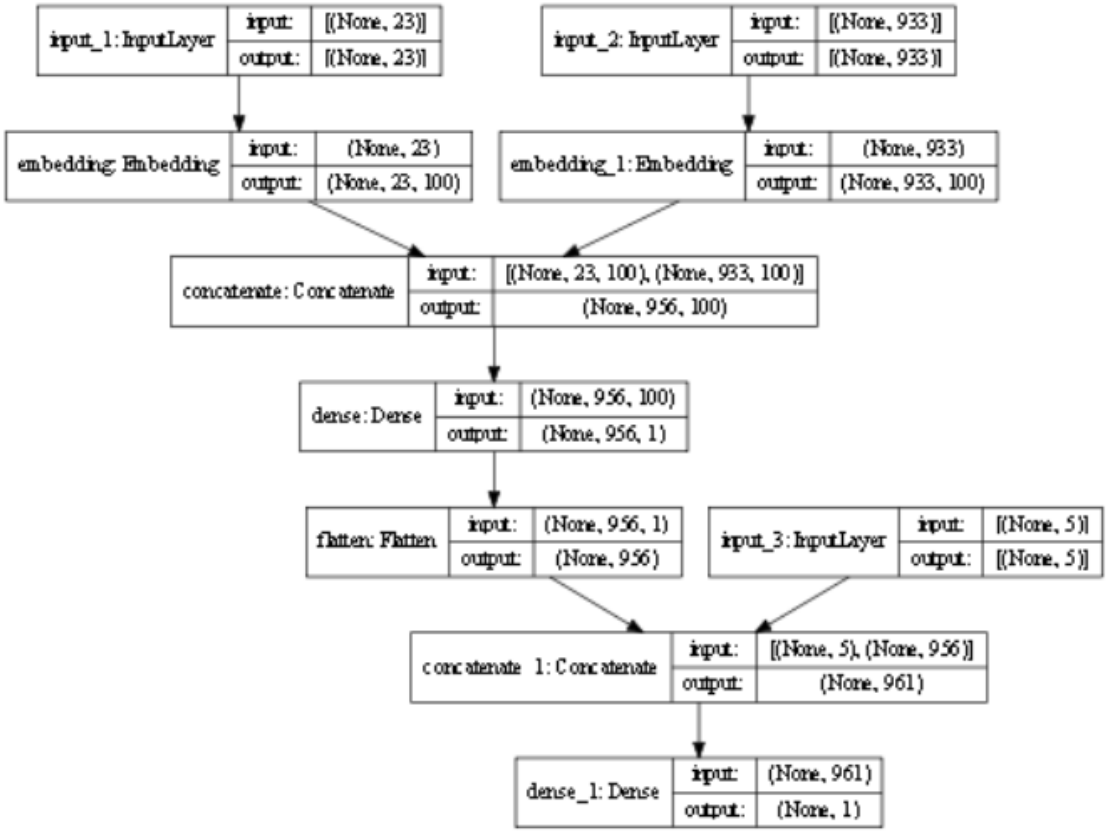


Figure 9: Architecture of Model for Experiment III

3.4.5 Experiment V - Simple Multi layer Perceptron with BERT Embedding

In this experiment we have used BERT embedding for title and description of the video. This model involves a multi layer perceptron which concatenates a BERT layer and meta data feature input layer. BERT embeddings are used to extract feature from the textual data. The key aspect of BERT is to apply a popular attention model called Transformer to language modeling. A Transformer includes an encoder and a decoder where encoder reads the text input and the decoder predicts the task. The advantage of using BERT as an embedding model is that it provides context-based representation for each word in a sentence. In contrast to this, Word2Vec provides representations which are fixed irrespective of where the word is used in the sentence. For example,

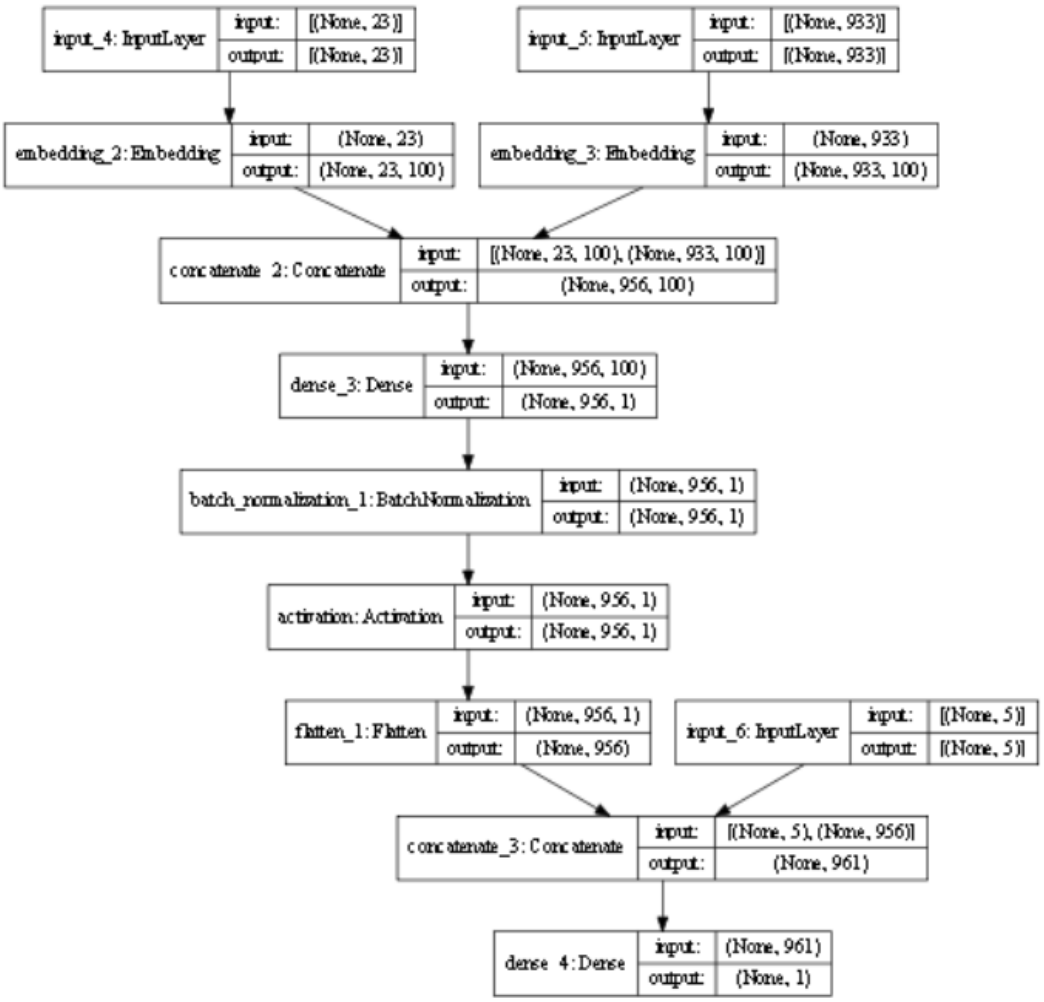


Figure 10: Architecture of Model for Experiment IV

in the sentences, ‘‘the man robbed the bank’’ and ‘‘The man was lying by the bank of the river’’, the word ‘‘bank’’ has different context. Word2Vec embeddings would give same numerical representation for the word ‘‘bank’’. BERT is a deeply bidirectional model The pretrained uncased model of BERT that is used in this experiment has 12 layers, 110M parameters and 768 hidden layers. The BERT tokenizer is used to split the words into tokens and returning inputIds and attention masks. Attention masks are used for padding. A mask value of one is for tokens that are not masked and a value zero means that the token is added by padding and should not be considered for

attention. This model takes input in some specific format, hence we appended special tokens ‘‘[SEP]’’, ‘‘[CLS]’’ to mark the end and beginning of the text. Each token is replaced with a inputId from the embedding table. The model uses Adam optimizer and the batch size is 10 for five epochs. We have used sequences of length 180 for this experiment. Figure 11 shows the model plotted. In this model, we have used a BERT embedding layer for textual data. The output of this embedding followed by a dense layer is then concatenated with the meta data features. After this, a dropout layer followed by a fully connected layer is used for classifying the data.

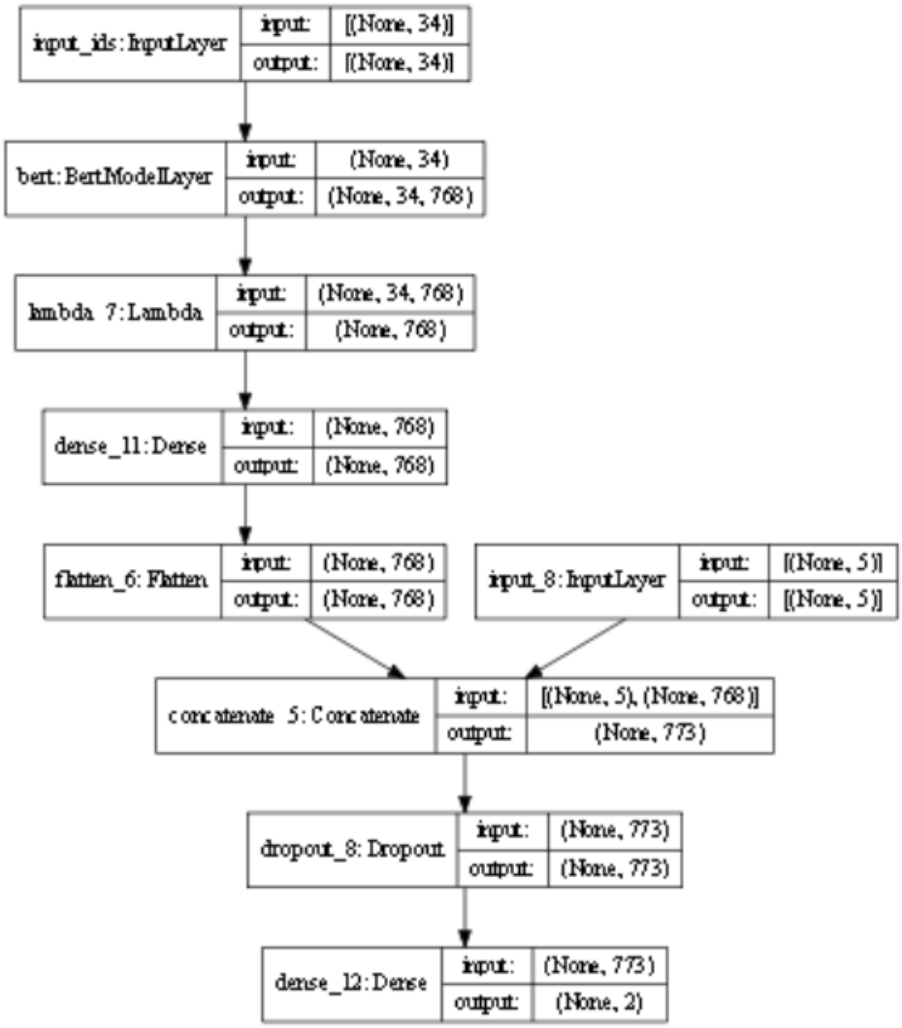


Figure 11: Architecture of Model for Experiment V

3.4.6 Experiment VI - DistilBERT Embedding

DistilBERT is a faster, lighter model and is a smaller variant of BERT Model. It runs 60% faster and has 45% less parameters than that of BERT [37]. We used DistilBert tokenizer for generating the inputIds and attention masks.For this experiment we have used pretrained distilbert-base-uncased model. The embeddings for tile and description are fed into a multi layer perceptron and is concatenated with the meta data features of the video and the YouTube channel. The model uses adam optimizer and the batch size is 10 for five epochs. Figure A.28 shows the model plotted. The architecture of this model has DistilBERT embedding for the textutal data. The input from the meta data features is concatenated followed by a dense layer for classification.

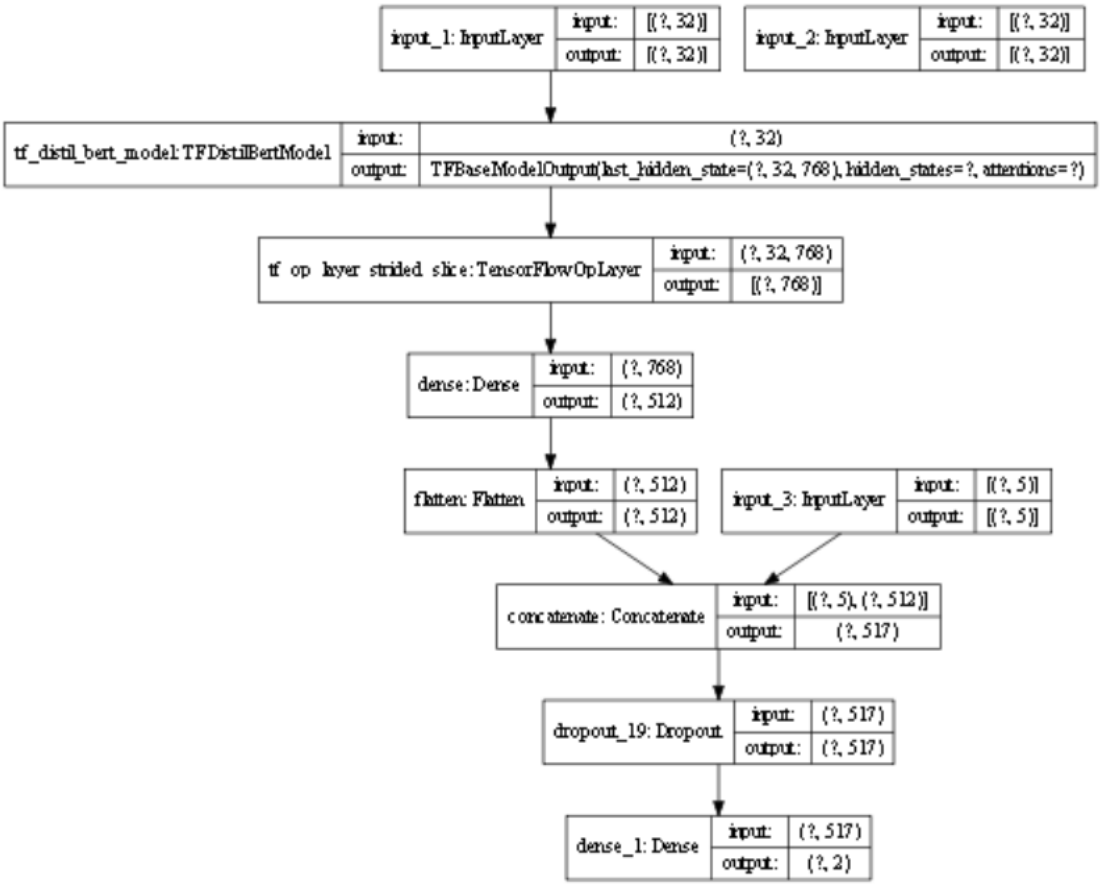


Figure 12: Architecture of Model for Experiment VI

CHAPTER 4

Results

Different architectures and techniques were implemented and trained throughout to experiment and achieve better accuracy. In experiment I, we have used a logistic regression with Word2Vec embeddings for features like title and description along with the meta data features. We get an accuracy of 52% with just title as input, and an accuracy of 70% with other features. This model is quicker to train and much easier to implement. Logistic regression will give a better accuracy for datasets which are more simpler. Figure 13 shows the ROC curve for the Logistic regression model.

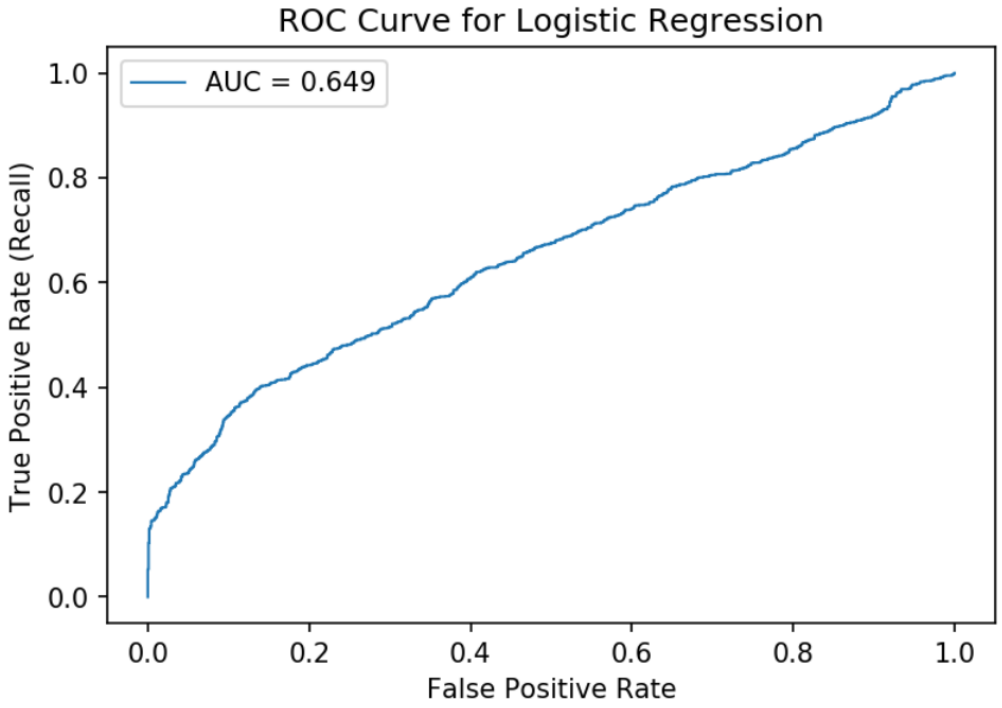


Figure 13: ROC Curve for Logistic Regression

Experiment II involved using a Random forest classifier on features like title, description, like count, dislike count, comment count, subscription count. We used Word2Vec embeddings for title and description. We trained this model in multiple

sets of input. The first set of input includes just the title and meta data features. The last set of input included all the features. We see that the accuracy improves as more features are added. In the first set of input we achieved an accuracy of 80%. The second set of input features gave an accuracy of 92%. Table 2 and Table 3 shows the classification report and accuracy for the model. Figure 14 shows the ROC curve for the Random forest model. Table 2 shows the precision and accuracy of 80.1% for the model with the first set of input as title and two meta data features for like count and dislike count.

Table 2: Classification Report for Random Forest Model with Input as Title, Like Count, Dislike Count

Class	Precision	Recall	F-score	Support
not-clickbait	0.81	0.80	0.81	1275
clickbait	0.80	0.81	0.80	1182
accuracy	---	---	0.80	2457
macro avg	0.80	0.80	0.80	2457
weighted avg	0.80	0.80	0.80	2457

Table 3 shows the report for the experiment II for input features like title and all the meta data features. The accuracy for this experiment is 92.5%. The report shows the precision and recall of the model in classifying clickbait and non-clickbait videos. The model performs slightly better in classifying non-clickbait videos more precisely.

Table 3: Classification Report for Random Forest Model with Input as Title, All Meta Data Features

Class	Precision	Recall	F-score	Support
not-clickbait	0.93	0.93	0.93	1275
clickbait	0.92	0.92	0.92	1182
accuracy	-	-	0.93	2457
macro avg	0.93	0.93	0.93	2457
weighted avg	0.93	0.93	0.93	2457

Figure 14 shows the ROC curve for the Random forest model where the input features included title, description and all the meta data features as like count, dislike count, comment count, subscription count, view count. The AUC score for this model is 0.95 which is pretty good with an accuracy of 94%. This shows that the model performs better in predicting true positives and false positives in the data. This experiment shows that adding more features increases the accuracy of the model.

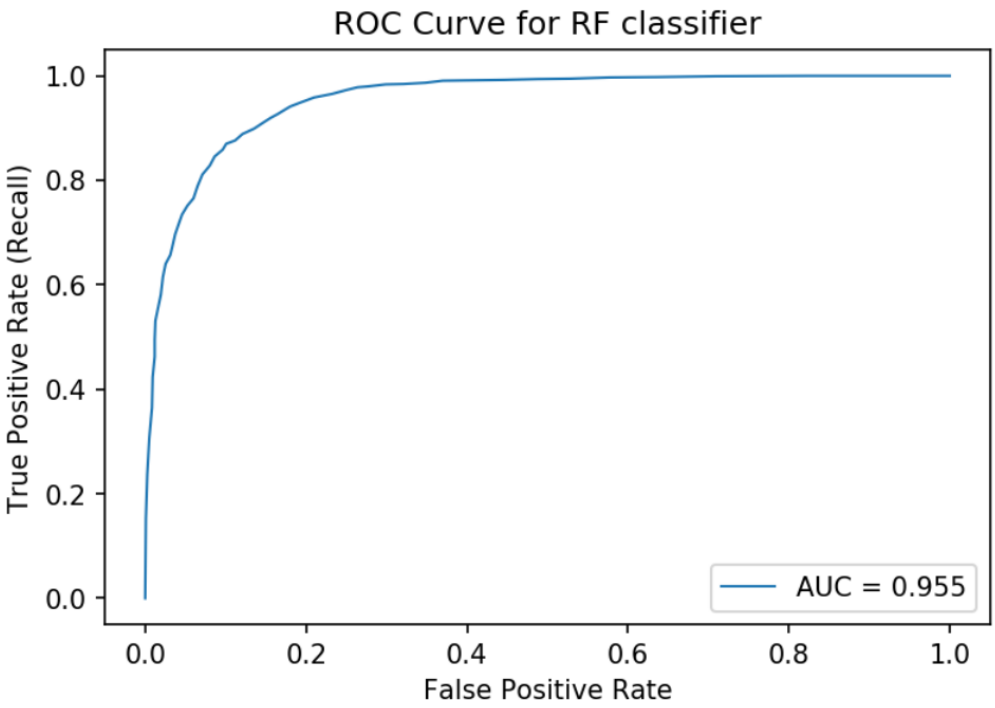


Figure 14: ROC Curve for Random Forest model with Input as Title, Description and All Meta Data Features

In experiment III, where a simple multi layer perceptron with Word2Vec embeddings for title and description are concatenated with meta data features, the accuracy observed is quite fluctuating in the training process and the average accuracy achieved is around 90%. Figure 15 shows the graph for accuracy over the number of training epochs. There is a huge dip in accuracy for epoch 26, however the accuracy increases significantly after that. The model was trained for 30 epochs. This can be because

some portion of the data is getting classified randomly leading to overfitting.

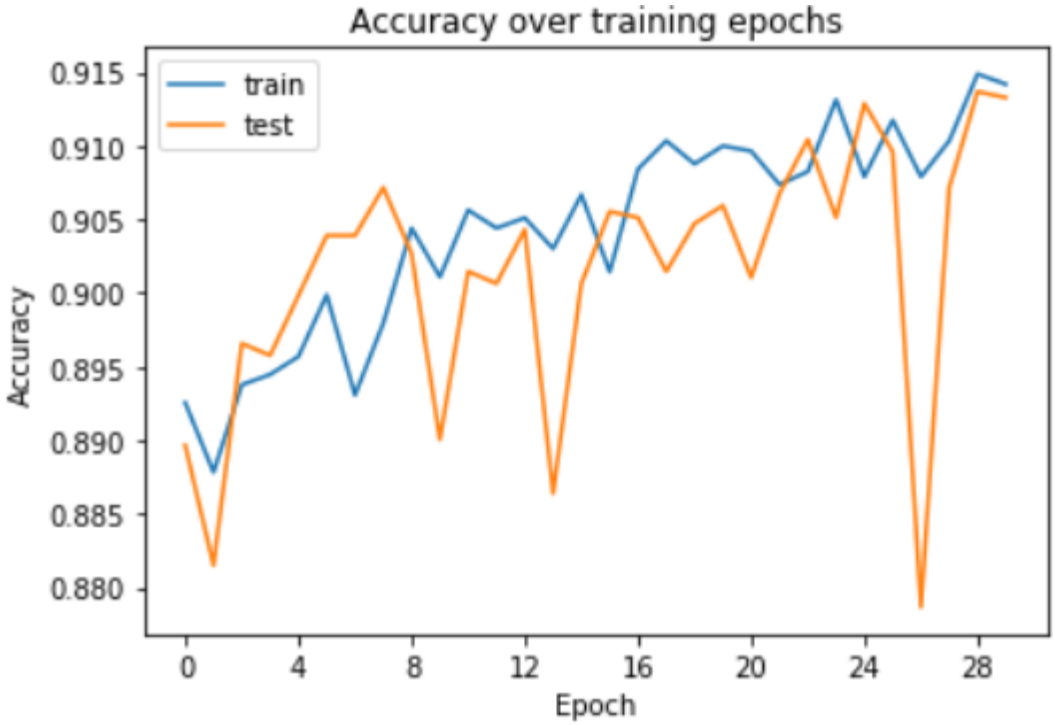


Figure 15: Accuracy Over Training Epochs for Experiment III Multi Layer Perceptron with Word2Vec

In experiment IV,where a modified multi layer perceptron is used with batch normalization and PReLU as an activation function, it has been observed that the accuracy increases to around 91%. In the previous experiment, the model seems to overfit as the gap between the validation accuracy and training accuracy is large. However in experiment IV, due to batch normalization the learning rates are higher and the network trains faster as well. Figure 16 shows the accuracy over the epoch graph for this experiment. The graph shows a stable increase in the accuracy without much fluctuations in the training and validation accuracy.

In experiment V, we have used a transfer learning model called BERT for word embeddings. BERT has proven to be efficient because of its deep contextualizing nature. Also, the results with the BERT gives an accuracy of 94.5%. In this experiment

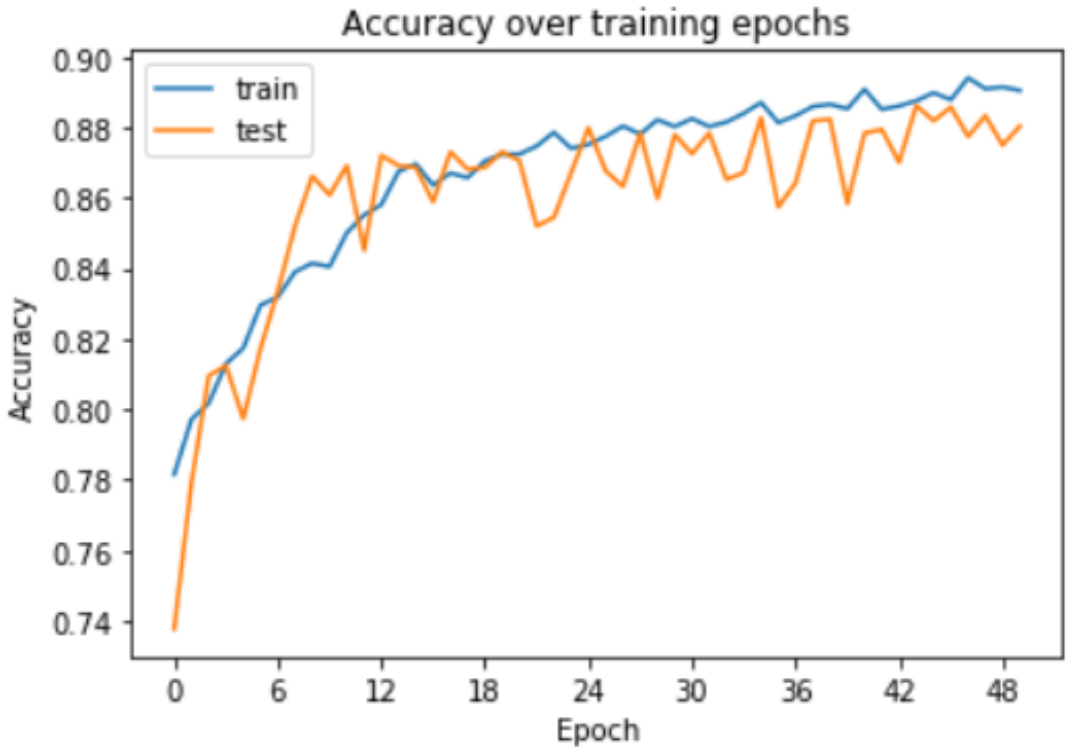


Figure 16: Accuracy Over Training Epochs for Experiment IV Multi Layer Perceptron with Word2Vec (BatchNorm and PReLU)

the length of input sequence is 180. Figure 17 shows the plot for accuracy over epoch graph for train and validation set. The graph shows a steep increase in the accuracy. However, in the last epoch the accuracy decreases and the loss increases to a small extent. Although, this model seems to perform quite well on the dataset. The time taken for training this model is 2 hours for 4 epochs for a sequence length of 180.

Figure 18 shows the plot for accuracy over epoch graph for validation set. The graph shows accuracy for different sequence lengths - 50, 80, 120, 150, 180, 200 for 3 epochs. The training time increases significantly as the sequence length increases. However, an increase in accuracy is also observed with increase in sequence length.

In experiment VI, we have used a lighter variant of BERT model for the word embeddings. The accuracy achieved is around 92%. This model is faster than the

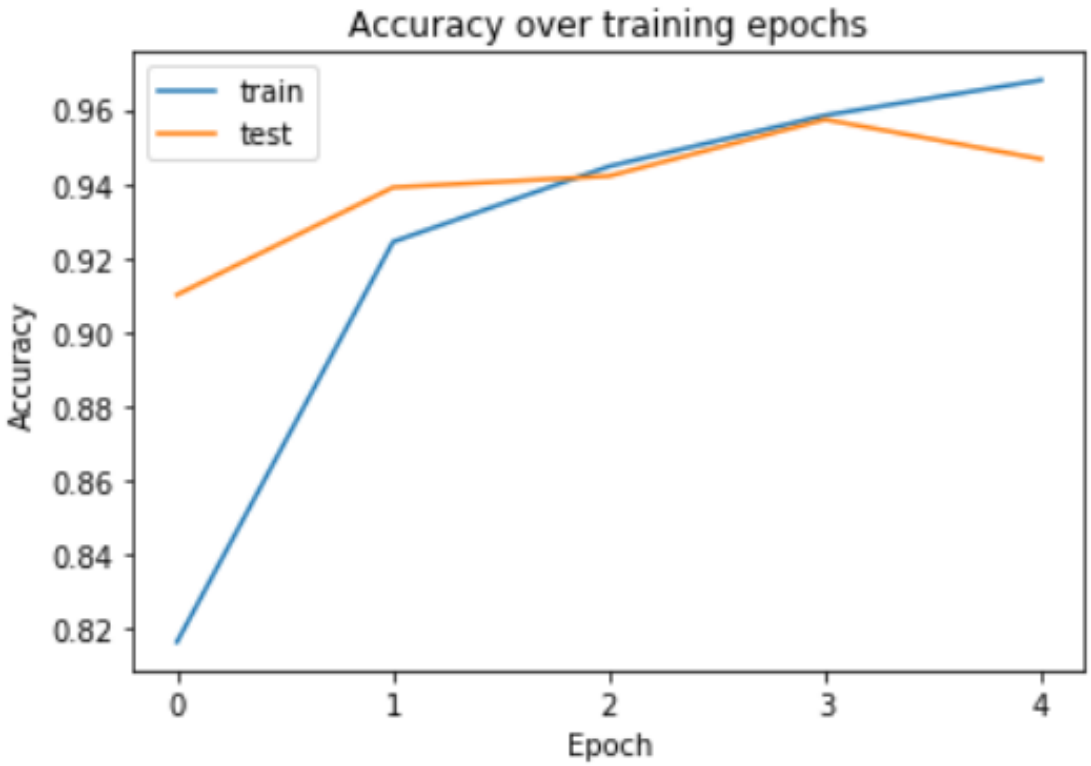


Figure 17: Accuracy Over Training Epochs for Experiment V Multi Layer Perceptron with BERT Embeddings

BERT and is good for smaller datasets. Although, DistilBERT is good in cases where smaller training time is important but the accuracy from BERT is better than DistilBERT. Table 4 shows the report for precision and recall for the clickbait and non-clickbait classifications. Figure 19 shows the plot for accuracy over epoch for the train and test set. After epoch two, there is a significant difference between the accuracy for the train and the test set. However, the accuracy increases thereafter.

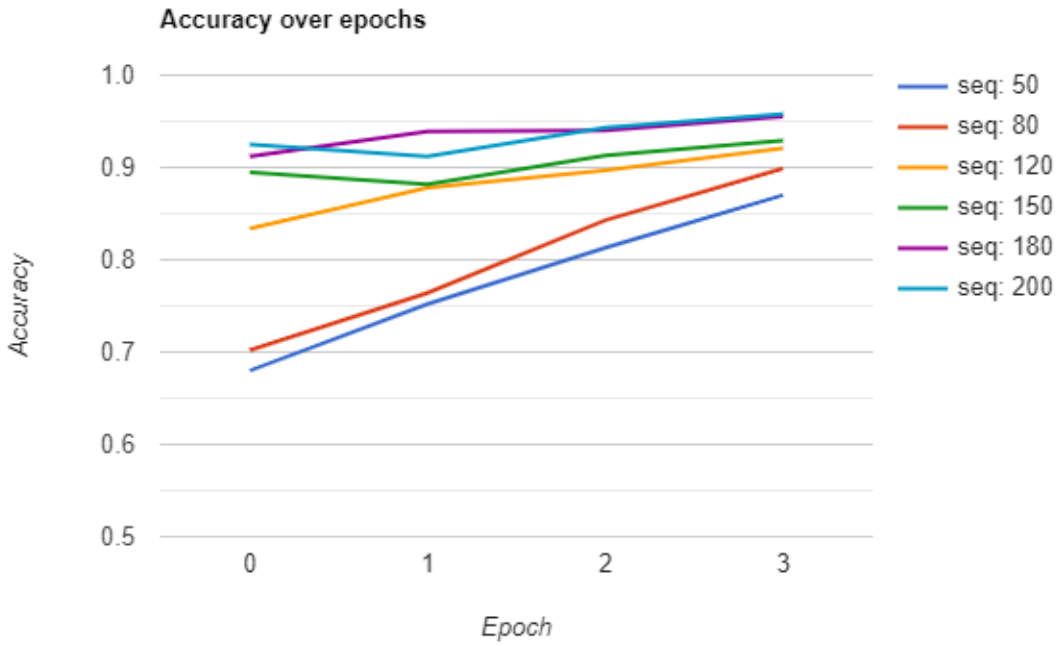


Figure 18: Test Accuracy Over Epochs for Experiment V Multi Layer Perceptron with BERT Embeddings for different sequence lengths

Table 4: Classification Report for Multi Layer Perceptron Model with DistilBERT

Class	Precision	Recall	F-score	Support
not-clickbait	0.92	0.95	0.93	884
clickbait	0.93	0.89	0.91	754
accuracy	-	-	0.92	1638
macro avg	0.92	0.92	0.92	1638
weighted avg	0.92	0.92	0.92	1638

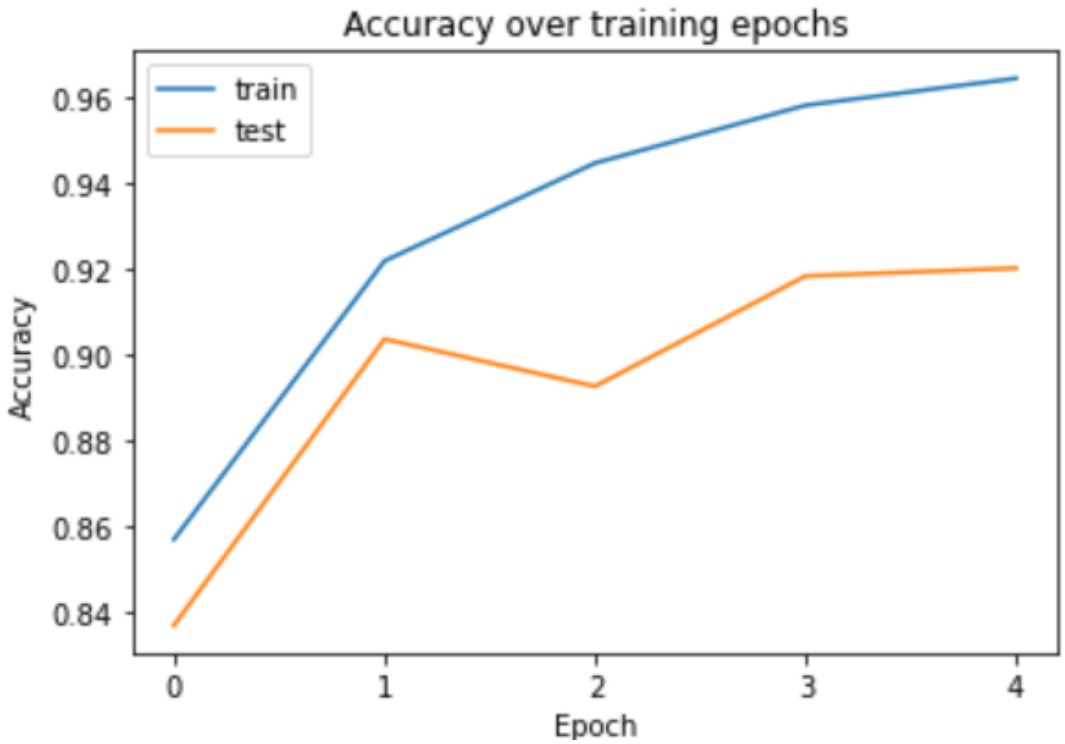


Figure 19: Accuracy Over Training Epochs for Experiment VI Multi Layer Perceptron with DistilBERT

CHAPTER 5

Conclusion and Future Works

This chapter discusses the conclusion and possible futures works for this project. A lot of research is going into detecting clickbaits in videos. Many more techniques and approaches can be utilized to extend this project.

5.1 Conclusion

The goal of this project is to utilize the state of the art techniques to classify a YouTube video if it is a clickbait or non-clickbait. A video has multiple parameters associated with it and each parameter contributes and has a property which can to some extent describe the video. We leverages three main types of features - user profile, video statistical features and textual data for the experiments. In this research, multiple experiments including logistic regression, random forest, multi layer perceptron with multiple inputs and multiple language modelings like BERT, Word2Vec, DistilBERT were employed. The best accuracy was achieved through the multi layer perceptron model using the BERT embeddings and the Random forest model. BERT embeddings have a contextualizing nature which worked really well in this project. Adding more features contributed to the accuracy of the models.

5.2 Future Work

In this project multiple experiments were performed on the dataset on features like title, description, meta data features like comment count, dislike count, like count, video duration, subscription count, view count. For the future works, more features of a video can be included. For example, the transcript of the video can be added as a feature as some videos are partially fake.These videos show some clickbait information as well as some credible information. Finding the cosine similarity between the transcripts and the title could give an information about their similarity. The network structure of the comments and replies which represents the semantic features

and attributes can also be constructed. This structure would help in analyzing user feedback for a video. Algorithms like Random Walk (RW) can be used to construct and capture features [2]. Image frames from the video can also be added as a feature to find the relevance of the frames with the title and the thumbnail.

Fact checking is another approach that can be employed to improve the accuracy. This includes validating the information using the credible sources. Information can be scrapped from some resources across Internet to verify whether a video is a clickbait or not [4]. This can be further enhanced by generating large dataset and comparing the accuracy of the models using the other publicly available dataset. Techniques like data augmentation and semi-supervised learning can be helpful .This process can also be made more robust by generalizing it across multiple platforms.

In this project we have experimented with BERT, Word2Vec, DistilBERT for word embeddings. Going further, we can experiment with DocToVec embeddings. In this project we used Random forest classifier which is an ensemble method based on decision trees. A gradient boosting algorithm called XGBoost which is another ensemble method can also be experimented. We can also experiment with Transformer-XL, which is a state of the art attentive language model. Also, XLNet which has been trained on a larger dataset than BERT and is also a bidirectional transformer can be used for word embeddings. XLNet uses a different approach than BERT in order to achieve bidirectional dependencies. It is also an extension of Transformer-XL for long term dependencies [38].

LIST OF REFERENCES

[1] ‘‘Clickbait,’’ Apr 2021. [Online]. Available: https://en.wikipedia.org/wiki/Clickbait

[2] L. Shang, D. Y. Zhang, M. Wang, S. Lai, and D. Wang, ‘‘Towards reliable online clickbait video detection: A content-agnostic approach,’’ *Knowledge-Based Systems*, vol. 182, p. 104851, 2019.

[3] J. Hennessey, ‘‘12 surprising examples of clickbait headlines that work,’’ Apr 2020. [Online]. Available: https://www.searchenginejournal.com/12-surprising-examples-of-clickbait-headlines-that-work/362688/#close

[4] D. Varshney and D. K. Vishwakarma, ‘‘A unified approach for detection of clickbait videos on youtube using cognitive evidences,’’ *Applied Intelligence*, pp. 1--22.

[5] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, ‘‘Stop clickbait: Detecting and preventing clickbaits in online news media,’’ in *2016 ieee/acm international conference on advances in social networks analysis and mining (asonam)*. IEEE, 2016, pp. 9--16.

[6] A. Elyashar, J. Bendahan, and R. Puzis, ‘‘Detecting clickbait in online social media: You won’t believe how we did it,’’ *arXiv preprint arXiv:1710.06699*, 2017.

[7] M. Glenski, E. Ayton, D. Arendt, and S. Volkova, ‘‘Fishing for clickbaits in social images and texts with linguistically-infused neural network models,’’ *arXiv preprint arXiv:1710.06390*, 2017.

[8] Y. Zhou, ‘‘Clickbait detection in tweets using self-attentive network,’’ *arXiv preprint arXiv:1710.05364*, 2017.

[9] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, ‘‘Fake news detection using machine learning ensemble methods,’’ *Complexity*, vol. 2020, 2020.

[10] A. Thota, P. Tilak, S. Ahluwalia, and N. Lohia, ‘‘Fake news detection: a deep learning approach,’’ *SMU Data Science Review*, vol. 1, no. 3, p. 10, 2018.

[11] H. Jwa, D. Oh, K. Park, J. M. Kang, and H. Lim, ‘‘exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert),’’ *Applied Sciences*, vol. 9, no. 19, p. 4062, 2019.

[12] D. Y. Zhang, L. Shang, B. Geng, S. Lai, K. Li, H. Zhu, M. T. Amin, and D. Wang, ''Fauxbuster: A content-free fauxtography detector using social media comments,'' in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 891--900.

[13] P. Palod, A. Patwari, S. Bahety, S. Bagchi, and P. Goyal, ''Misleading metadata detection on youtube,'' in *European Conference on Information Retrieval*. Springer, 2019, pp. 140--147.

[14] K. S. Reddy, K. S. Nihith, M. S. Chowdary, and T. K. Prasad, ''An efficient word embedded click-bait classification of youtube titles using svm,'' in *Symposium on Machine Learning and Metaheuristics Algorithms, and Applications*. Springer, 2019, pp. 175--184.

[15] M. Dong, L. Yao, X. Wang, B. Benatallah, and C. Huang, ''Similarity-aware deep attentive model for clickbait detection,'' in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2019, pp. 56--69.

[16] A. R. Setlur, ''Semi-supervised confidence network aided gated attention based recurrent neural network for clickbait detection,'' *arXiv preprint arXiv:1811.01355*, 2018.

[17] E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro, ''Some like it hoax: Automated fake news detection in social networks,'' *arXiv preprint arXiv:1704.07506*, 2017.

[18] B. Zaman, A. Justitia, K. N. Sani, and E. Purwanti, ''An indonesian hoax news detection system using reader feedback and naïve bayes algorithm,'' *Cybernetics and Information Technologies*, vol. 20, no. 1, pp. 82--94, 2020.

[19] S. Kumar, R. West, and J. Leskovec, ''Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes,'' in *Proceedings of the 25th international conference on World Wide Web*, 2016, pp. 591--602.

[20] A. M. Turing, ''Computing machinery and intelligence,'' in *Parsing the turing test*. Springer, 2009, pp. 23--65.

[21] P. C. Rajapaksha and R. W. Vidanelage, ''Clickbait detection using multimodel fusion and transfer learning,'' Ph.D. dissertation, Institut polytechnique de Paris, 2020.

[22] A. Graves, ''Generating sequences with recurrent neural networks,'' *arXiv preprint arXiv:1308.0850*, 2013.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ''Efficient estimation of word representations in vector space,'' *arXiv preprint arXiv:1301.3781*, 2013.

[24] M. Saifee, ‘‘Recent advancements in nlp (1/2),’’ Dec 2019. [Online]. Available: https://medium.com/swlh/recent-advancements-in-nlp-1-2-192ac7eefe3c

[25] L. Weng, ‘‘Learning word embedding,’’ Oct 2017. [Online]. Available: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

[26] I. Sutskever, O. Vinyals, and Q. V. Le, ‘‘Sequence to sequence learning with neural networks,’’ *arXiv preprint arXiv:1409.3215*, 2014.

[27] D. Bahdanau, K. Cho, and Y. Bengio, ‘‘Neural machine translation by jointly learning to align and translate,’’ *arXiv preprint arXiv:1409.0473*, 2014.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ *arXiv preprint arXiv:1706.03762*, 2017.

[29] Maxime, ‘‘What is a transformer?’’ Mar 2020. [Online]. Available: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

[30] Khalid, ‘‘Bert explained: A complete guide with theory and tutorial,’’ Sep 2019. [Online]. Available: https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/

[31] A. Agrawal, ‘‘Logistic regression. simplified.’’ Mar 2017. [Online]. Available: https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389

[32] T. Yiu, ‘‘Understanding random forest,’’ Aug 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[33] Franckepeixoto, ‘‘A simple overview of multilayer perceptron (mlp) deep learning,’’ Dec 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/

[34] ‘‘Conda.’’ [Online]. Available: https://docs.conda.io/en/latest/

[35] S. Zannettou, S. Chatzis, K. Papadamou, and M. Sirivianos, ‘‘The good, the bad and the bait: Detecting and characterizing clickbait on youtube,’’ in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 63--69.

[36] Alessiovierti, ‘‘alessiovierti/youtube-clickbait-detector.’’ [Online]. Available: https://github.com/alessiovierti/youtube-clickbait-detector

[37] ‘‘Distilbert.’’ [Online]. Available: https://huggingface.co/transformers/model_doc/distilbert.html

[38] P. Tum, ‘‘A survey of the state-of-the-art language models up to early 2020,’’ Nov 2020. [Online]. Available: https://medium.com/@phylypo/a-survey-of-the-state-of-the-art-language-models-up-to-early-2020-aba824302c6

APPENDIX

Graphs for the Experiments

input_1 (InputLayer)	[(None, 23)]	0	
input_2 (InputLayer)	[(None, 933)]	0	
embedding (Embedding)	(None, 23, 100)	992200	input_1[0][0]
embedding_1 (Embedding)	(None, 933, 100)	5267900	input_2[0][0]
concatenate (Concatenate)	(None, 956, 100)	0	embedding[0][0] embedding_1[0][0]
dense (Dense)	(None, 956, 1)	101	concatenate[0][0]
input_3 (InputLayer)	[(None, 5)]	0	
flatten (Flatten)	(None, 956)	0	dense[0][0]
concatenate_1 (Concatenate)	(None, 961)	0	input_3[0][0] flatten[0][0]
dense_1 (Dense)	(None, 1)	962	concatenate_1[0][0]

Total params: 6,261,163
Trainable params: 1,063
Non-trainable params: 6,260,100

Figure A.20: Summary for Experiment III - Simple Multi Layer Perceptron with Word2Vec Embeddings

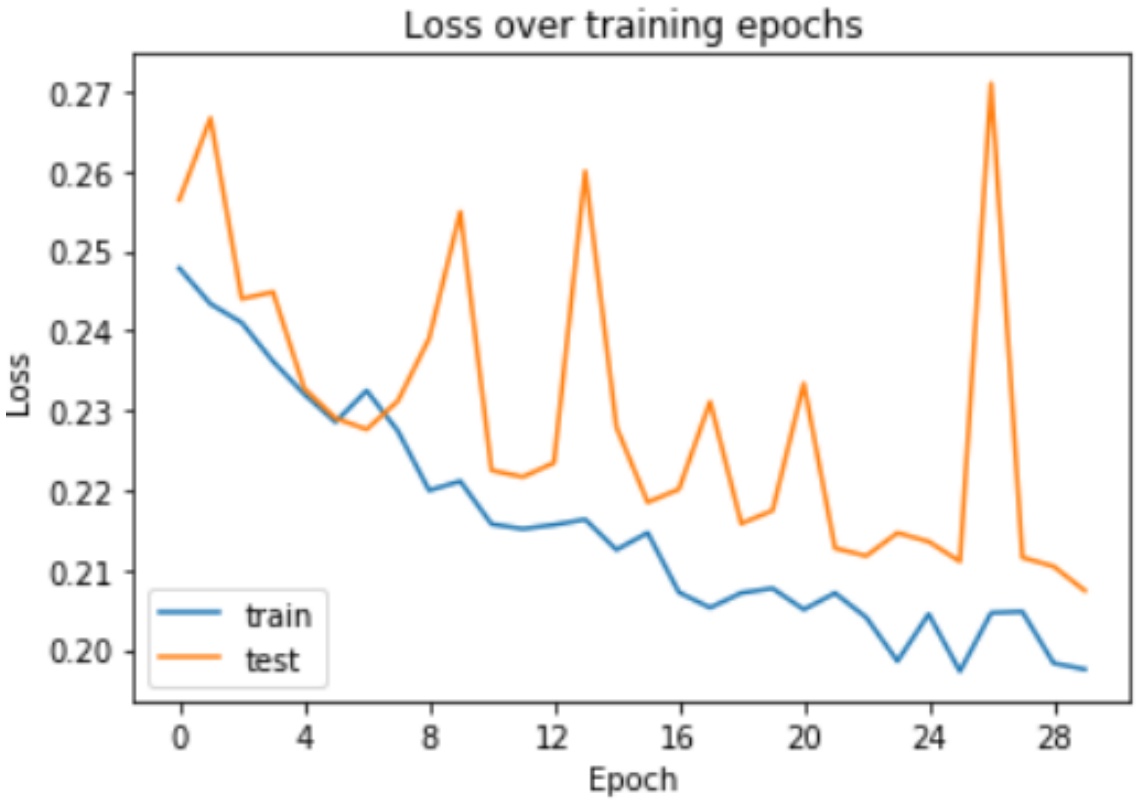


Figure A.21: Loss Over Epoch for Experiment III - Simple Multi Layer Perceptron with Word2Vec Embeddings

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	[(None, 23)]	0	
input_5 (InputLayer)	[(None, 933)]	0	
embedding_2 (Embedding)	(None, 23, 100)	992200	input_4[0][0]
embedding_3 (Embedding)	(None, 933, 100)	5267900	input_5[0][0]
concatenate_2 (Concatenate)	(None, 956, 100)	0	embedding_2[0][0] embedding_3[0][0]
dense_3 (Dense)	(None, 956, 1)	101	concatenate_2[0][0]
batch_normalization_1 (BatchNor	(None, 956, 1)	4	dense_3[0][0]
activation (Activation)	(None, 956, 1)	0	batch_normalization_1[0][0]
input_6 (InputLayer)	[(None, 5)]	0	
flatten_1 (Flatten)	(None, 956)	0	activation[0][0]
concatenate_3 (Concatenate)	(None, 961)	0	input_6[0][0] flatten_1[0][0]
dense_4 (Dense)	(None, 1)	962	concatenate_3[0][0]

Total params: 6,261,167
Trainable params: 1,065
Non-trainable params: 6,260,102

Figure A.22: Summary for Experiment IV - Simple Multi Layer Perceptron with Word2Vec Embeddings (BatchNorm + PReLu)

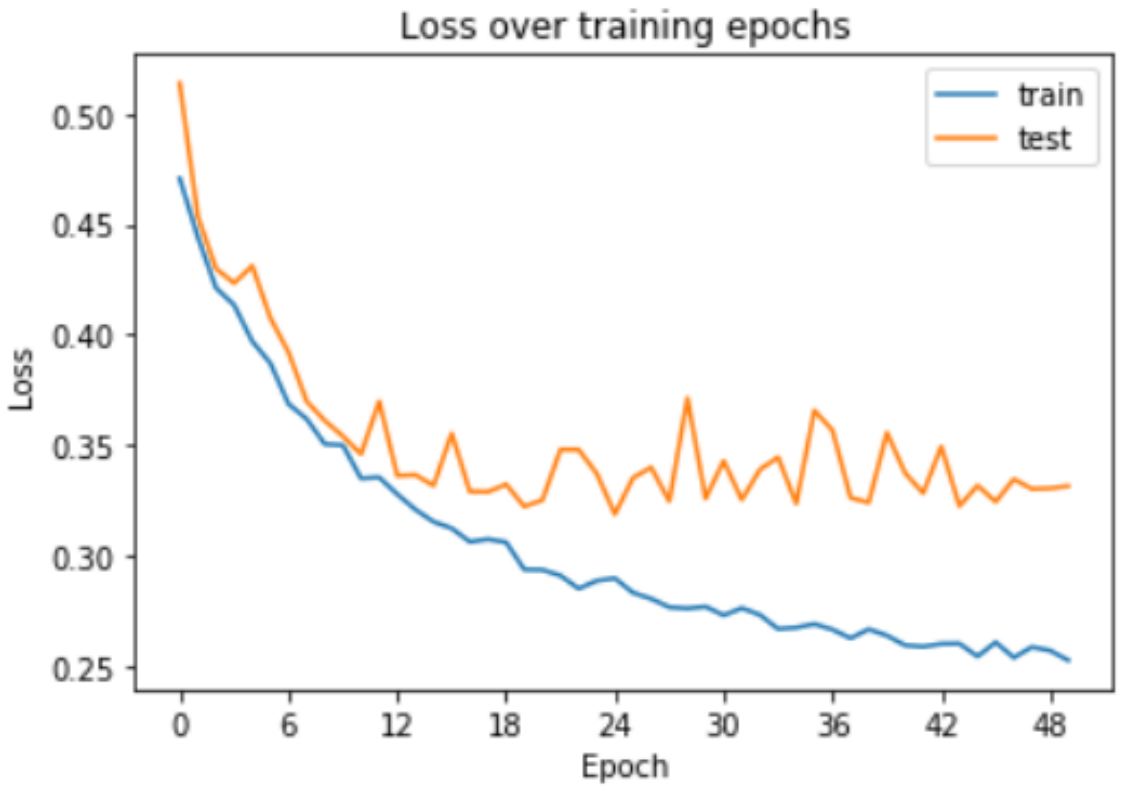


Figure A.23: Loss Over Epoch for Experiment IV - Simple Multi Layer Perceptron with Word2Vec Embeddings (BatchNorm + PReLu)

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 34)]	0	
bert (BertModelLayer)	(None, 34, 768)	108890112	input_ids[0][0]
lambda_1 (Lambda)	(None, 768)	0	bert[0][0]
dense_2 (Dense)	(None, 768)	590592	lambda_1[0][0]
input_2 (InputLayer)	[(None, 5)]	0	
flatten_1 (Flatten)	(None, 768)	0	dense_2[0][0]
concatenate_1 (Concatenate)	(None, 773)	0	input_2[0][0] flatten_1[0][0]
dropout_1 (Dropout)	(None, 773)	0	concatenate_1[0][0]
dense_3 (Dense)	(None, 2)	1548	dropout_1[0][0]

Total params: 109,482,252
Trainable params: 109,482,252
Non-trainable params: 0

Figure A.24: Summary for Experiment V - Multi Layer Perceptron with BERT

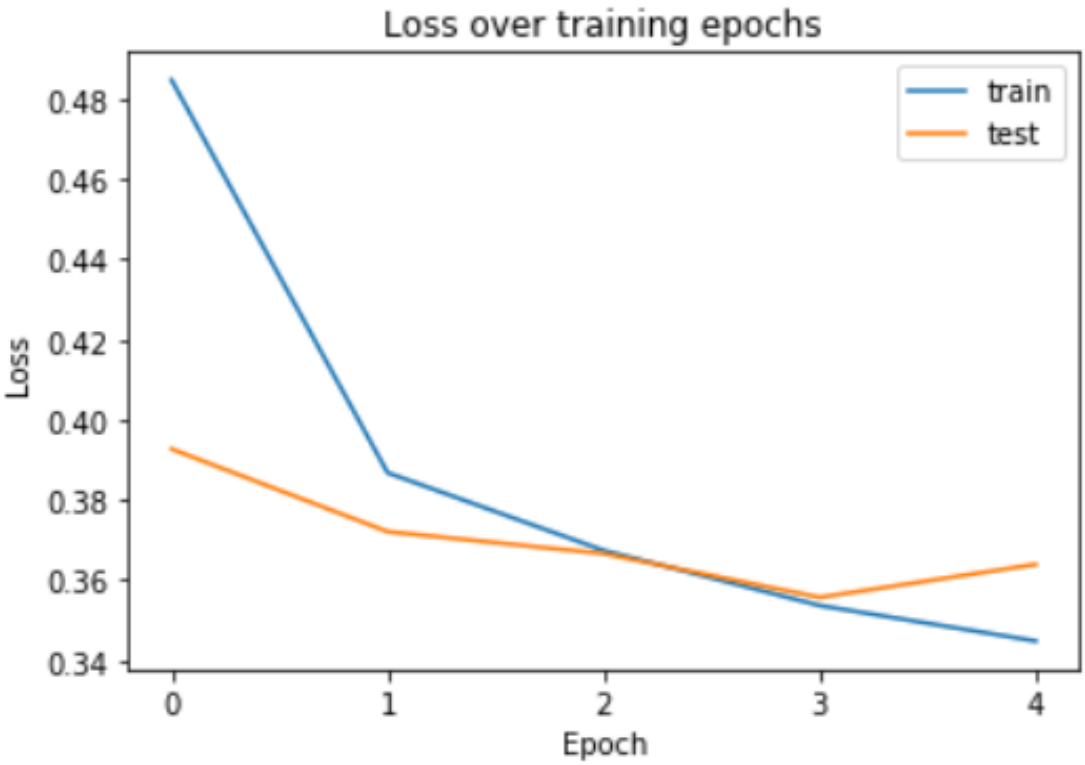
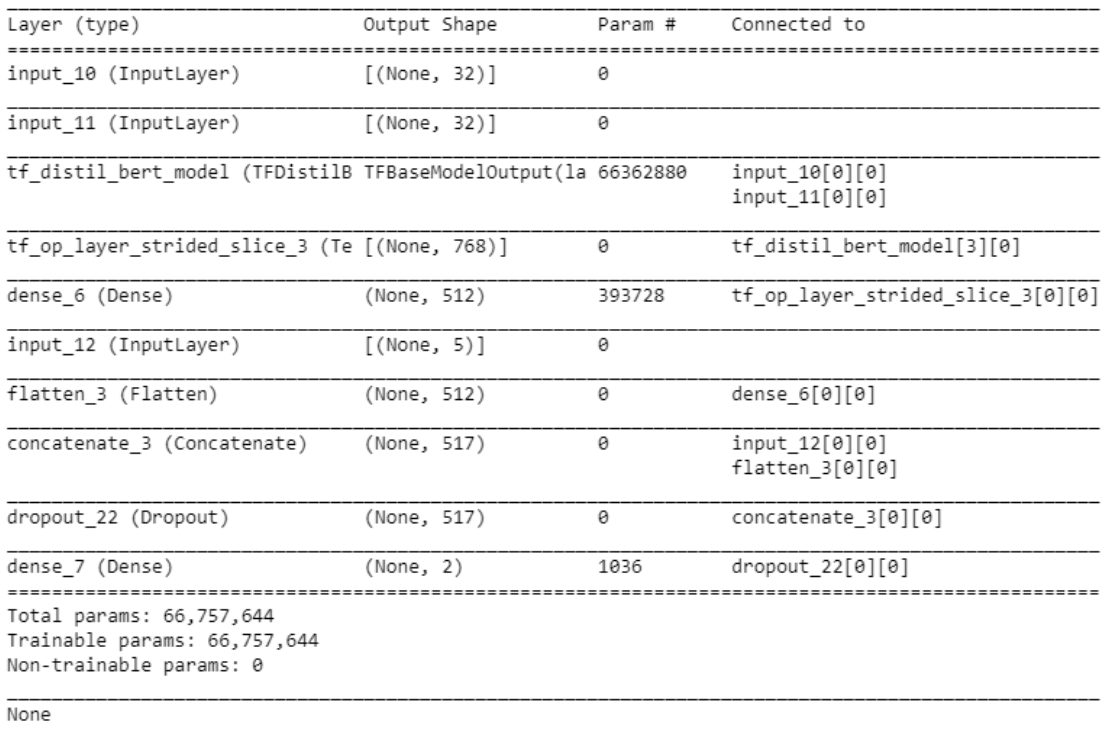


Figure A.25: Loss Over Epoch for Experiment V -Multi Layer Perceptron with BERT



Layer (type)	Output Shape	Param #	Connected to
input_10 (InputLayer)	[(None, 32)]	0	
input_11 (InputLayer)	[(None, 32)]	0	
tf_distil_bert_model (TFDistilB	TFBaseModelOutput(la	66362880	input_10[0][0] input_11[0][0]
tf_op_layer_strided_slice_3 (Te	[(None, 768)]	0	tf_distil_bert_model[3][0]
dense_6 (Dense)	(None, 512)	393728	tf_op_layer_strided_slice_3[0][0]
input_12 (InputLayer)	[(None, 5)]	0	
flatten_3 (Flatten)	(None, 512)	0	dense_6[0][0]
concatenate_3 (Concatenate)	(None, 517)	0	input_12[0][0] flatten_3[0][0]
dropout_22 (Dropout)	(None, 517)	0	concatenate_3[0][0]
dense_7 (Dense)	(None, 2)	1036	dropout_22[0][0]

Total params: 66,757,644
Trainable params: 66,757,644
Non-trainable params: 0

None

Figure A.26: Summary for Experiment VI - Multi Layer Perceptron with DistilBERT

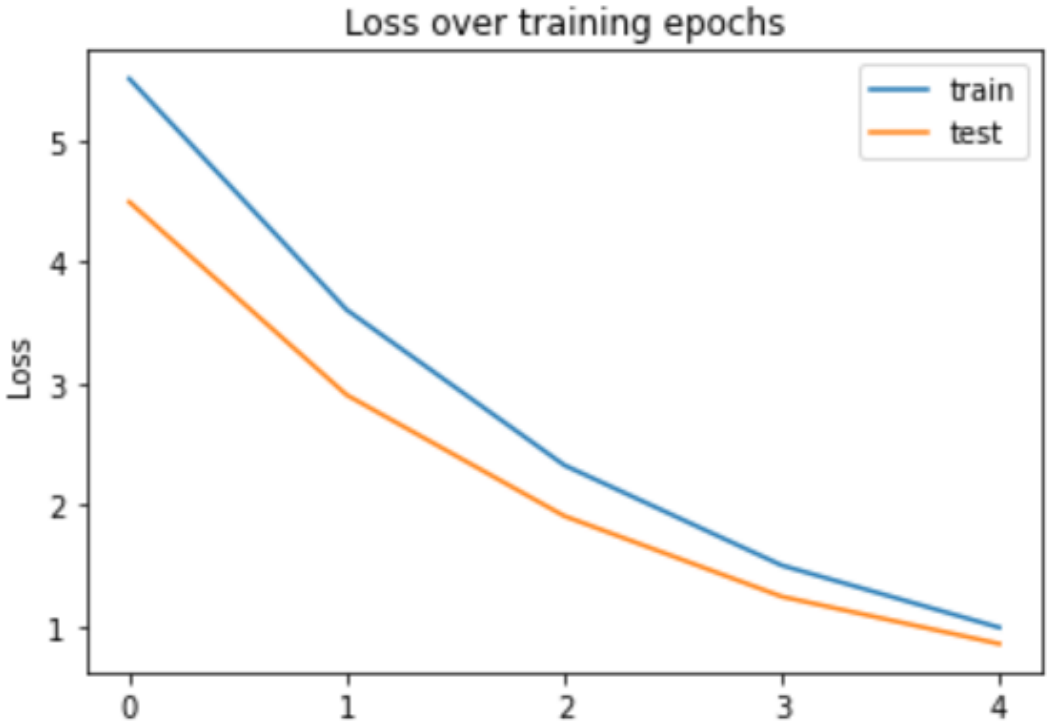


Figure A.27: Loss Over Epoch for Experiment VI - Multi Layer Perceptron with DistilBERT

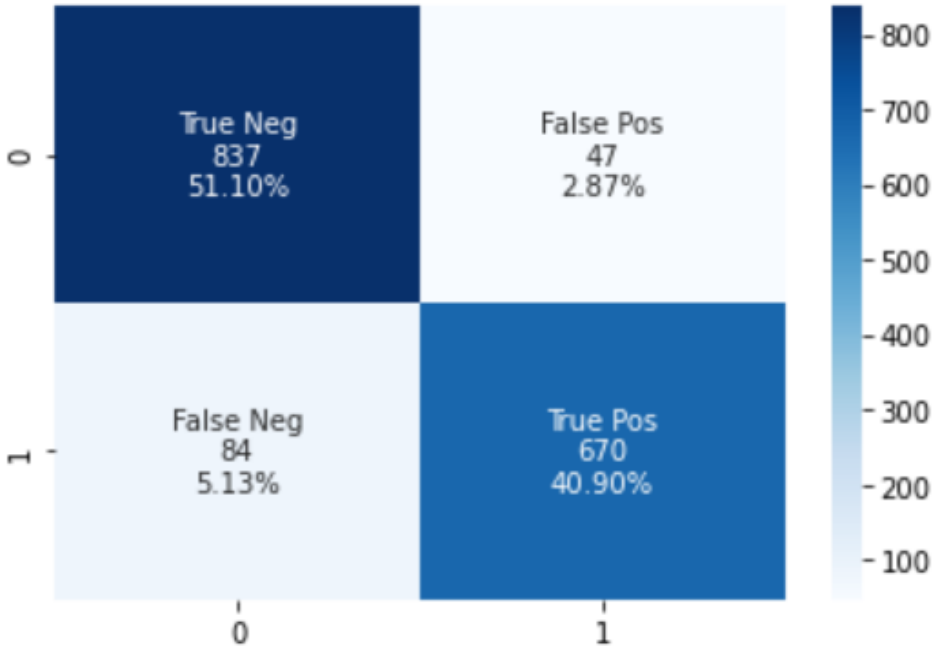


Figure A.28: Confusion Matrix for Experiment VI - Multi Layer Perceptron with DistilBERT