ANALYSIS OF CLICKBAIT IN YOUTUBE VIDEOS USING ENSEMBLE MODELS

A PROJECT REPORT

Submitted by

VISHAL R. K.

VISHAL KRISHNAN S. H.

Under the guidance of

Mr. V. Balaji

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

B.S. ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE & TECHNOLOGY

(Estd. u/s 3 of the UGC Act. 1956)

www.crescent.education

JULY 2021

B.S. ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE & TECHNOLOGY

(Estd. u/s 3 of the UGC Act. 1956)

www.bsauniv.ac.in

BONAFIDE CERTIFICATE

Certified that this project report "**ANALYSIS OF CLICKBAIT IN YOUTUBE VIDEOS USING ENSEMBLE MODELS"** is the bonafide work of **"VISHAL R. K. (170071601143) and VISHAL KRISHNAN S. H. (170071601142)"** who carried out the project work under my supervision. Certified further, that to the best of our knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr. V. BALAJI
SUPERVISOR
Assistant Professor
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

SIGNATURE

Dr. E. SYED MOHAMED
HEAD OF THE DEPARTMENT
Professor & Head
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

B.S. ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE & TECHNOLOGY

(Estd. u/s 3 of the UGC Act. 1956)

www.bsauniv.ac.in

VIVA VOCE EXAMINATION

The viva voce examination of the project work titled **"ANALYSIS OF CLICKBAIT IN YOUTUBE VIDEOS USING ENSEMBLE MODELS"**, submitted by **VISHAL R. K. (170071601143) and VISHAL KRISHNAN S. H. (170071601142)** is held on _____

SIGNATURE

Mr. V. BALAJI
SUPERVISOR
Assistant Professor
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

SIGNATURE

Dr. E. SYED MOHAMED
HEAD OF THE DEPARTMENT
Professor & Head
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

ACKNOWLEDGEMENT

We sincerely express our heartfelt gratitude to **Dr. A.PEER MOHAMED,** Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology, for providing us an environment to carry out our course successfully.

We sincerely thank **Dr. A. AZAD**, Registrar, for furnishing every essential facility for doing our project.

We thank Dr. Venkatesan Selvam, Dean, School of Computer, Information and Mathematical Sciences for his motivation and support.

We thank **Dr. E. SYED MOHAMED,** Professor and Head, Department of Computer Science and Engineering, for providing strong oversight of vision, strategic direction and valuable suggestions.

We obliged to our internal guide **Mr. V. Balaji,** Designation, Department of Computer Science and Engineering for her professional guidance and continued assistance during our project.

We express our sincere thanks to the Project Review Committee members, from the Department of Computer Science and Engineering, **Dr. Sharmila Sankar,** Professor**, Dr. L. Arun Raj,** Associate Professor and **Mrs. R. Akila,** Assistant Professor (Sr. Grade) for their valuable suggestions and support.

We thank our class advisor, **Mrs. R. Akila,** Assistant Professor (Sr.Grade), Department of Computer Science and Engineering for her guidance and encouragement throughout the project period.

We thank all the **Faculty members** and the **System Staff** of Department of Computer Science and Engineering for their valuable support and assistance at various stages of project development.

VISHAL R. K.

VISHAL KRISHNAN S. H.

TABLE OF CONTENTS

ABSTRACT

Clickbait is one of the most prevalent problem of the twenty-first century internet. It is the simple act of enticing the viewers with a catchy and attractive headline or cover, only for the user to be sorely disappointed to find what was being advertised is not actually in the content. Or worse, the content contains blatant lies and misinformation which leads to the spread of hoaxes and scams all over the internet, be it via forums, group chats, or even through word of mouth.

Normally, clickbait as a research topic is primarily applied in news articles and text-based content. But we are trying to apply the same principles into the world of video-based content and the largest video-based content host and provider on the internet is YouTube. In this age of intolerance and impatience, people are quick to believe what they see, especially if it is negative. No one has the composure to fact-check what they see. Moreover, video-based content consumption is much more appealing to the younger generation compared to text-based resources. Thus, it is imperative that such an impressionable audience must not be fooled by what media they consume.

In this project, we are analyzing the extent of clickbait on YouTube, which categories are more at fault compared to others, and try to develop a classification ensemble model that can detect if a video is bait or not by dissecting the various components and features of a YouTube video. We will make use of state-of-the-art technologies to parse the various media features and try to come up with our own classification model that considers user feedback to tie it all up.

Keywords: Clickbait, YouTube, Neural Networks, Word Embeddings, Image Captioning, Social Media Mining, Web Scraping

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview

Click baits are techniques used by various content creators to lure the user to their content by promising something that is not in their content. Click baits are one of the most commonly found phenomena now found on the internet which consumes the user's time for something they do not desire. Clickbait detection is something that has not been given any attention by the platforms where the content is broadcasted mainly because click baits usually draw more attention with their misleading titles and description than content which stays true to its information.

One of the major leading platforms for the consumption of content is YouTube. YouTube contains a very vast library of videos that almost covers any topic that you could think of. Every domain, sub-domain, the topic would have a YouTube video on it. Being a platform for videos, the click baits are more prevalent in YouTube when compared to other platforms, this is because YouTube consists of videos and only videos, which are harder to analyze and quantify with relevance to what was promised.

1.2 Description

The motivation behind click baiting is nothing too deep, it is just monetary gain. YouTube makes its revenue primarily in a similar manner to Google AdSense – an advertising agency that Alphabet Inc. owns that helps post advertisements on online web pages based on the user who is using said site. YouTube had generated a revenue of nearly $20 billion USD just in 2020, a marked 33% increase over 2019 and some YouTubers like Mr Beast and Dude Perfect earn over $20 million through just YouTube AdSense. Since YouTube content creators (or YouTubers) earn money through ad impressions, which in turn is affected by video clicks, watch time, and viewer retention rate, it is clear that YouTubers tend to stall their main subject at

hand, which is posted in their title and thumbnail to tedious lengths just to make sure that the viewer has not clicked off of the video yet. Clearly, most YouTubers would be pushed to waste the user's precious time by stalling the content, and since YouTube is a site with nearly 2 billion users, most of which are kids, teens, and young adults, it becomes evident that such sinister business practices paint a bleak picture of the world and its future to the target audience.

1.3 Objective

Our goal here is to analyze the YouTube videos and to identify if a particular video is legitimate or if it is just baiting users into watching the videos while not providing what the users desire. To do so, we first need a way for the computer system to understand what is happening in the video, and not just the numbers which make up the video. To achieve this, we have different approaches for different kinds of videos. We need to categorize the videos into two main classes which define the kind of video it is,

- Dialogue driven video
- Action driven video

Dialogue-driven videos are videos where the content or the information from the video is mainly in the form of dialogues or speech, to understand these kinds of videos, we first convert the audio in the video to a transcript, analyze the resultant transcript.

Action-driven videos are videos where the content of the information is transmitted visually to the users, these kinds of videos cannot be approached the same way as we approached dialogue-driven videos as these kinds of videos could have minimal to no dialogues in them. To analyze these videos, we use a model which describes the events happening in the video. Once we get the description of what is happening in the videos, we can do further analysis on the video.

Once we analyze the video itself, we take a look at the attributes of the video, such as likes, dislikes, comments, views, etc., to get an even more

clear picture regarding the video's veracity. These are prime factors in analyzing the video because a majority of these attributes consist of the opinions of other users, and as we all know, whether a YouTube video is clickbait or not can be extremely subjective at times. Of course, there are some offences that are certainly clickbait, but there is a lot of grey area as well. This is where user feedback becomes extremely handy. Besides, the thought of clickbait stems from the notion of the users being misled in the first place.

2. RELATED WORK

Clickbait has existed in various forms over the past couple of decades, starting with the ever-so old newspapers, which have now turned into news websites. Getting the correct information from this world of over sensationalized journalism is difficult. People are enticed by the attractive pictures, and headlines, and subtitles, only to be left sorely disappointed that the content of the article wasn't worth their time. This has been explored further by various others before, but one that caught our eye was [1]. They claim that huge amounts of revenue are generated with those clicks and ad impressions and it is not going to stop unless the users are aware of the litany of misinformation littered throughout these articles. Some of the most commonly used techniques in click bait articles include uppercase letters, over exaggeration of facts, and sometimes even lies. These click bait headlines generally allude to, but never obviously state what the subject matter of the article is all about. A news articles whose subject matter is as simple as, say, "Benefits of Yoga", would be click baited as "Your friends are doing THIS to get in SHAPE!!!". Sure, yoga will get a person into shape, but never has it been stated in the title of the article that it is about yoga, and not about something that can achieve similar results. It is imperative to understand clickbait from both sides of the argument – the business side and the consumer side. Companies use clickbait because they evidently benefit from visits to their site, as their company runs on views and clicks on their advertisements, who provide money to sustain said website. The problem is, their gain is the consumer's loss – loss of high quality, reliable, and concise source of information. This creates a hole in the space of journalism. Users start avoiding articles because they think that all of them are clickbait, and companies start assuming that the articles aren't click bait enough and try to widen the gap between reality and exaggeration even more. To prevent from such a divide occurring, a clickbait detection system is of utmost necessity.

Another important factor that [1]., mentions is – context. The role of context in any form of measurement of truthfulness is undervalued in the current age of informational outrage and spontaneity. People of the current generation rarely read into an article that is longer than 100 words, let alone 500 or 1000. All they

need are the headlines and a picture or elsewhere known as a thumbnail, and that is it. this can be noticed in the uprise of various short media formats – TikTok, Moj, TakaTak for short video format social media, InShort and DailyHunt for short news media format and so on. This would also explain why most articles on the web mention the average reading time of each article – the attention span of the current and future generations is extremely short. Therefore, they tend to ignore context to most things, especially news. They take the headlines, interpret it in their sense, and actualize it within themselves as facts. We cannot force them to read the entire article as well. Thus, the clickbait detection systems in place must make an unconditional use of context and interpret it with as easy a metric as possible, in terms of understandability, and the one that they chose is – percentages. To make use of the context of the articles in question, they developed a Recurrent Convolutional Neural Networks (RCNN) to parse misdirecting content titles. RCNN is prepared for safeguarding and catching the more extended context of the headlines. RCNN catches the left and right setting of a sentence with the usage of a Bi-directional RNN. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) can also be used in place of the typical RNN unit and they have found out that a combination of RCNN with GRUs produced the most accurate result in the tests, with an accuracy of 97.76%. nevertheless, this metric must only be taken with a grain of salt since this does not consider the entire article in question, only the headlines. Our takeaway from their research is that RCNN+GRU is great starting point to make use of when it comes to clickbait detection, and also to consider other factors apart from the headline only.

Web scraping is an interaction of separating significant and fascinating content data from website pages. The greater part of the current examinations focusing on this undertaking are generally about robotized web information extraction. In the extraction cycle, these examinations initially make a DOM tree and afterward access the fundamental information through this tree. The development cycle of this tree builds the time cost contingent upon the information design of the DOM Tree. In the current web scraping writing, it is seen that time effectiveness is overlooked. Our research makes comprehensive use of web scraping and we looked towards [2], to make use of

a much more efficient algorithm for scraping the web, especially when it comes to time constraints. They have developed a library called UzunExt that follows a two-pronged approach to extract content from the DOM of a site. We will make use of this library in conjunction with BeautifulSoup to improve the efficiency of web scraping.

Humans are exceptional in the world of identifying things in pictures. We can discern objects from background, facts from fiction and abstract things as well. This is quite the conundrum because YouTube thumbnails play an influential role in alluring viewers to click on a video. We can easily identify the techniques used by the content creator to captivate users, but the problem of programmatically identifying the baiting nature of a thumbnail is challenging to say the least. The approach that we are going to put into use is one that is proposed in [4]. Most of what has been achieved in the world of image captioning and naming pictures with a fixed arrangement of visual classifications and extraordinary advancement has been accomplished in these undertakings. Notwithstanding, while shut vocabularies of visual ideas comprise a helpful displaying presumption, they are immeasurably prohibitive when contrasted with the colossal number of rich depictions that a human can create. Some spearheading approaches that address the test of producing picture portrayals have been created. In any case, these models regularly depend on hard-coded visual ideas and sentence layouts, which forces limits on their assortment. Additionally, the focal point of these works has been on lessening complex visual scenes into a solitary sentence, which is actually something contrary to what we are endeavouring to accomplish. We need as detailed a description as possible and then extract the necessary info from said description. Restricting to a single sentence might leave out key information that may be the difference between click bait and not clickbait.

The essential test towards this objective is in the plan of a model that is adequately rich and at the same time reason about substance of pictures and their portrayal in the area of regular language. Moreover, the model ought to be liberated from suppositions about explicit hard-coded formats, rules or classifications and rather depend on gaining information from the images themselves. The second test is that datasets of picture inscriptions are

accessible in huge amounts on the web however these portrayals multiplex notices of a few elements whose areas in the pictures are obscure. One way to train using such a dataset, and one that we will be putting to use, is to treat the sentences as weak labels. This way, the neural network would then try to optimize itself and thus, tend to generate more than what it actually should intend to, given such a dataset to train with. In this way, we can create a semi-supervised neural network i.e., one that is trained using a dataset but also generative in nature.

Although other facets of the research are challenging in their own right, one that truly stumped us was the problem of parsing the video and generating a summary for said videos. There have been attempts made in the past to summarize videos, but they largely rely upon gathering the most important frames of the video and treat them as individual images to caption them. Unfortunately, that ignores the most important rule of clickbait detection – considering context.

[5], propose a novel end-to-end sequence-to-sequence model to generate captions for videos. they make use of LSTMs in recurrent neural networks to leverage image captioning technologies into the video world. Unlike the thumbnail description part of the research, this can only generate sentences. Moreover, to mitigate the fact that we simply do not have the computing power to even transfer learn such a gargantuan model, we make use of another technique – Video Summarization. Using the process proposed by [11] we can summarize long videos to short 30 second snippets that preserve most of the information of the video. Besides, it makes sense to use such a technique to emulate humans as we also just scrub through long videos to get to the part that we want [8] ‘s LSTM model is trained on video-sentence pairs and learns to associate a sequence of video frames to a sequence of words in order to generate a description of the event in the video clip. They evaluate several variants of the model on a standard set of YouTube videos and two movie description datasets (M-VAD and MPII-MD).

One of the major bottlenecks that we faced during the design and ideation phase of this research is this: we are able to generate a brief description for the

thumbnail, extract the title and description from YouTube's Public Data and Analytics API and, provided the resources, we can generate a simple summary for an entire video as well. But how do we even compare or find similarity across such a disparate spread of features, let alone feed them into a learning model. that is where S-BERT or Sentence BERT comes into play. S-BERT is an adaptation of BERT, the famous semantic textual similarity (STS) assessing model, except it makes use of Siamese and triplet networks to prepare sentence embeddings that can then be used in conjunction with a similarity metric like Jaccard similarity, cosine similarity, Manhattan distance etc… Here, they make use of cosine similarity and, by processing sentence embeddings beforehand, they are able to perform STS in just 10-15 seconds over a 10,000-sentence corpus, which would take days for the vanilla version of BERT. Simply put, we are making use of a state-of-the-art sentence embedding model to create vector word embeddings of our model. Do note that a pre-trained SBERT model like paraphrase-mpnet-base-v2 will give a 768-vector for each sentence. this is still highly infeasible at the current scale. Besides, it would not make any sense to feed all of the values to the model because we humans do not use all of these to discern a clickbait video from a non-clickbait one. We compare these metrics and see if they have any overlap. For example, we check whether subject matter of the title or the thumbnail of the video is being discusses in the comments. This is the motivation behind our decision to include the cosine similarity (recommended by SBERT) over the actual values and the creator fed values. this also reduces the dimensions of the input features by multiple orders of magnitude, thanks to SBERT.

3. PROBLEM DEFINITION AND METHODOLOGIES

3.1 Problem Definition

The problem that we are tackling can be described in simple words as follows:

We are trying to analyze a subset of YouTube videos and discern how click bait is being deployed in order to deceive viewers. Then, using the knowledge and insight gained, we are trying to develop a model that can separate and identify clickbait and non-clickbait videos without the intervention of the user (if possible).

In order to make matters clear, following are some prominent pointers that can be used to cognitively determine whether a video might be clickbait or not, and also a way to show how semantically difficult the problem is.

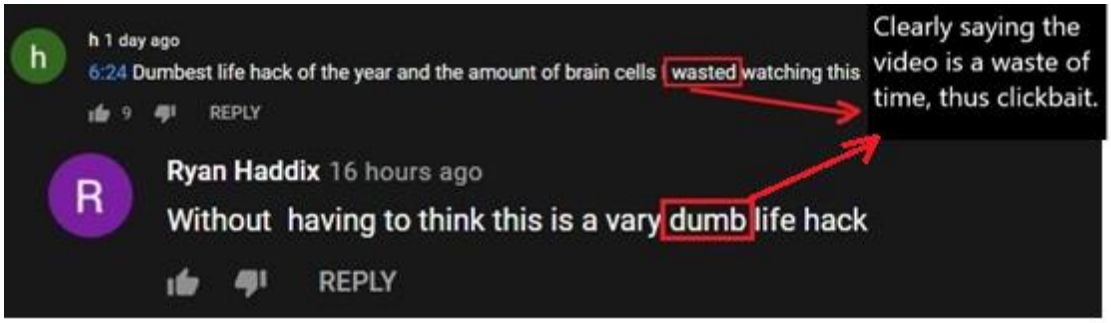

Figure 3-1 Users complaining about clickbait

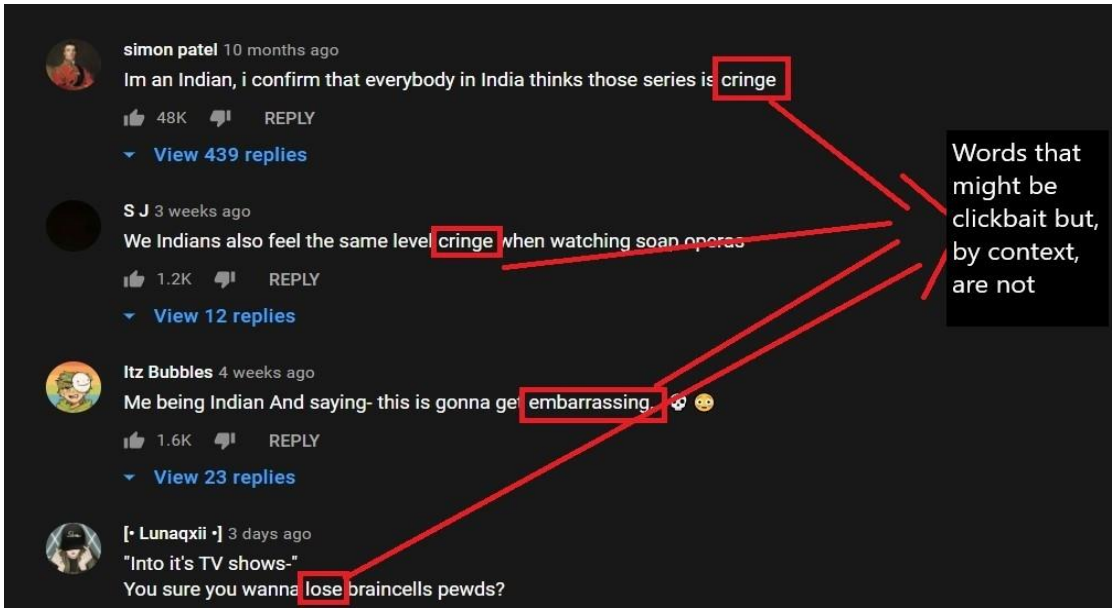

Figure 3-2 Clickbait-denoting words used outside of the context of clickbait

Figures 3-1 and 3-2 show two contrasting examples of similar content. On the one hand, words that denote clickbait (i.e., fake, dumb, waste of time, lose time, cringe, etc…) are clearly used by commenters to denote their displeasure about the video in question, as in Figure 3-1.

On the other hand, the same words can be used to describe the content within a video (it may be a reaction or a commentary channel), and the comments are directed at that content withing the video and not the uploader or the video itself. this may sound extremely confusing to a person who has never watched a reaction or a commentary channel before, so I suggest to look up a few to see the difference, as it is easier to experience than to explain.

So, we can't entirely rely on user comments. What about the likes and dislikes? That cannot be a singular metric to determine the clickbaity-ness of a video, simply because the viewers might just be displeased with the video and might have disliked it because they don't like the content. That does not necessarily mean that the user has been click-baited.

Therefore, the question of selecting the features to determine clickbait within a YouTube video is in and of itself no easy feat. We, however arrived at a simple solution. While examining over 450+ videos, we found a simple pattern – Clickbait videos tend to have way less engagement than a non-clickbait video.

Of course, this feature standalone cannot be used to determine clickbait (extremely dedicated fanbases of YouTubers can just like and comment on a video before they even watch one second of it), but a combination of like-dislike ratio, similarity of the comments and the content of the video and viewer engagement with the video can be definitely used to gauge the clickbaity-ness of a video.

Figure 3-3 showcases a video that is uploaded by a YouTube channel called 5-Minute Crafts, which has over 72 million subscribers. That being said, just glancing at this one picture, without even the thumbnail, we can see what is wrong:

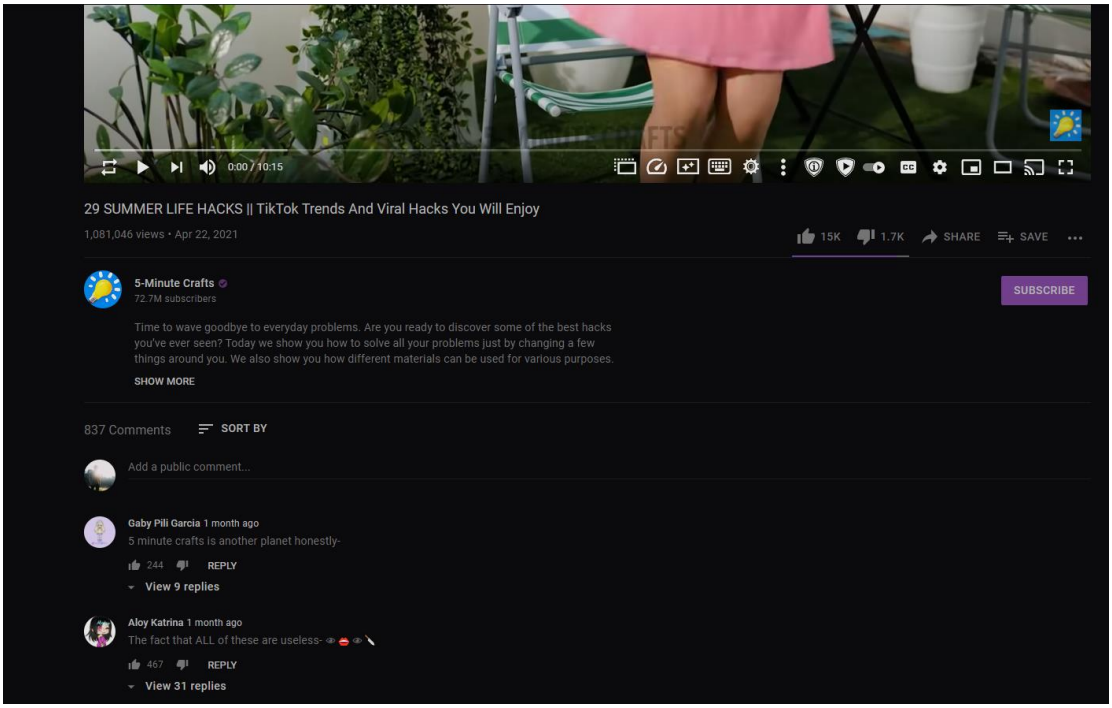

Figure 3-3 A sample clickbait video

The following point detail every single feature that point it towards being clickbait:

1. For a 72M subscriber channel, only 1M viewers have watched.
2. Even out of that, only 17.5K people have interacted with the video, this shows that most people clicked off of the video very early on.
3. The top comments say that most of the content in the video is useless.
4. The title uses current trend buzzwords like TikTok, Trends, etc…
5. Finally, as a nail in the coffin, here is the thumbnail of the video:

Figure 3-4 Thumbnail of a typical clickbait video

Safe to say that the video that is being discussed is not very true to its title or thumbnail.

3.2 Existing System

There are no existing models being used by viewers to identify clickbait in YouTube videos as this is an evolving field that is prime for research and research only. Therefore, the existing system can be established as “viewers guessing whether a video is clickbait or not”.

Clearly there are a host of disadvantages tied to such a system. Some of them are listed below:

1. Uneducated viewers or users unfamiliar to the digital world basically are rolling a dice every time they open a new video hoping to watch what was advertised in the title and the thumbnail.
2. Even educated viewers or viewers who are well versed in the ways of clickbait still have to waste their time masking educated guesses by considering various factors. Some of which are glancing through the comments, look at the number of likes and dislikes, the length of the video, the user who uploaded the video and so on, all of which eat up so much time that, more often than not, the user just gives up and watches the video only to be disappointed.
3. There are clever tricks that the uploaders perform in order to escape the click bait nature of their videos. these tricks range from being mildly sinister to utterly devious. This includes the likes of enticing the viewers with an attractive female, only for her to be in the video for less than 10 seconds, outright lying about helping the homeless people or charities, or even faking a celebrity’s demise. The list goes on and it will never stop as long as it works in favor of the uploader
4. There might be false information or news that is being shared in a click bait video and users who watch it may think it is real, and start spreading it to their group chats and this leads to a massive hoax or a scam. Misinformation has always been a threat to the society, and given the current state of the world we live in, it is inhumane to do such a thing, yet money matters.

3.3 Proposed System

Our proposed system consists of a trained ensemble learning model that makes use of deep and classification learning techniques and uses the video's metadata and other features to identify if it is clickbait or not.

This approach is highly advantageous because of the following reasons:

1. It requires little to no input from the user, all we need is the video ID and that is it.
2. It considers various features that includes, but is not limited to, video views, likes, dislikes, thumbnail, audio transcript, title, comments, and the uploader's details.
3. It also checks if there are any news articles about the subject matter of the video just to make sure that there are no hoaxes.
4. Since it is a trained model, it will not foster any biases towards a particular creator on the platform, and the only biases that it may have are the people who label the videos clickbait are not.

There are various algorithms that aid us in this research. We will be discussing the most prominent ones.

3.4 Algorithms

3.4.1 Speech Recognition

The first algorithm that we see being used is a speech recognition model in order to extract the sentences to obtain the dialogue driven aspect of the YouTube video.

Now, in order to make things clear, SpeechRecognition is a Python library that contains various online and offline APIs to perform speech recognition, it is not a model in and of itself. It is an extremely helpful tool nonetheless. The model that we used however is Google's Cloud Speech-To-Text model. It is LSTM-GRU based, but it is a proprietary and billable resource, so we do not have extensive details about it.

3.4.2 Sent2Vec – SBERT

Senc2Vec is actually SBERT which we discussed in our Related work. Unlike most NLP algorithms, it is neither RNN nor CNN based. It is entirely based upon the BERT model, which in turn heavily bases itself off of the Transformer model in Vaswani et. al. it is a self-attention model, which is extremely simple compared to the other neural network architectures.

Transformers are dominating the field of NLP as of late thank to works like BERT and GPT3. Self-attention layers are extremely easy to explain. For each word in a sentence, a^i, three values are calculated, namely Query, Key and Value, each of which have a weight attached to them which are the parameters of the model. Using the dot multiplication of the query and key values of each word, an attention matrix is created (which is divided by the square root of the dimension in order to account for variance.) Now, the sigmoid function is applied over the attention matrix, and all values of each row are multiplied to the actual value of the word calculated earlier to calculate b^i, which, together with all b's gives us the vector representation of the sentence. The advantage of such a system is that it is not time series based, like LSTMs, so it can be performed parallelly. And since the calculations can be boiled down to matrices, it doesn't even need tensors in most cases. Thus, it is the best, and most efficient model to represent sentences. This has been improved upon by BERT and SBERT in more ways than one.

3.4.3 RCNN – Ensemble Model

There is no specific name to these models, but RCNNs are generally used to transform a problem from computer vision to natural language. this is typically used in captioning and summarization of pictures and videos, that is the exact reason why it is being used here as well. It is called an ensemble model because it makes use of two different learning models (RNN and CNN).

Figure 3-5 shows the architecture of the RCNN that we made use of, to caption YouTube thumbnails.

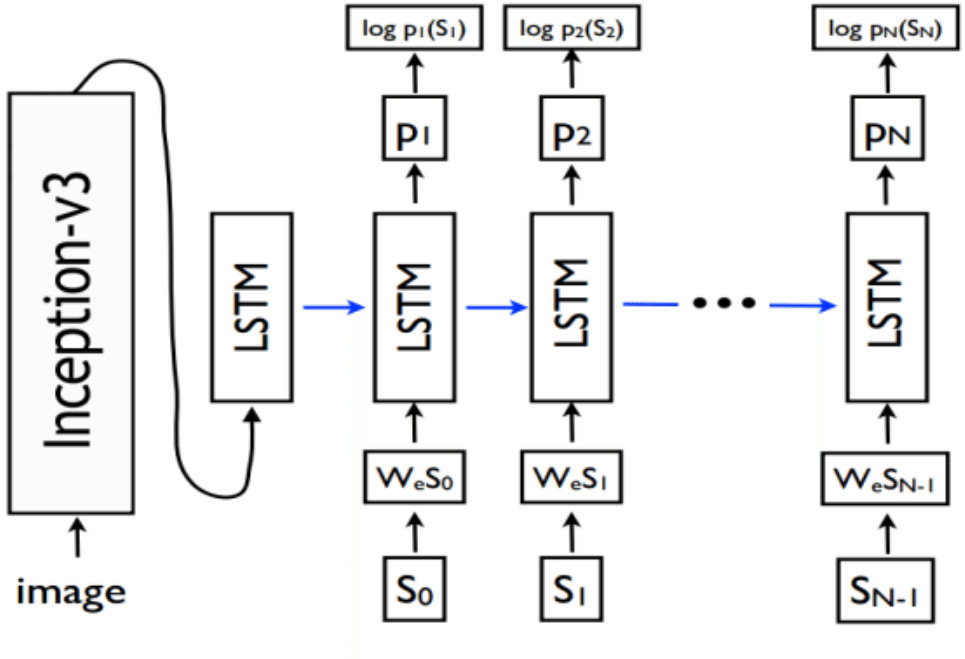

Figure 3-5 Show and Tell - Architecture (Courtesy: arxiv.org)

The CNN in the model is a pre-trained version of the famous InceptionV3 model developed by Google. It is an extremely deep CNN making use of parallel CNN layers and contacting the average of the output of the parallel layers, a short circuit layer that acts as an auxiliary classifier near the end and many more oddities.

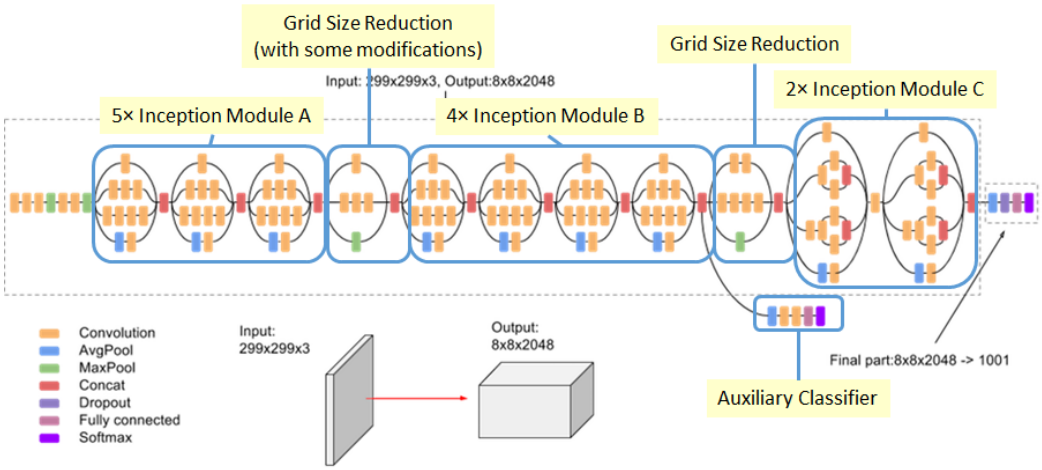

Figure 3-6 InceptionV3 Architecture

Fortunately, we won't have to write and train this from scratch as it would take literal days to complete training. Instead, we can make use of the pre-trained weights to this model available online. The LSTM however we had to train it from scratch using a standard set of pictures and captions dataset available with the Show and Tell paper.

To understand how LSTMs work, we need to get a grasp of how RNNs work. An RNN's working can be explained as follows:

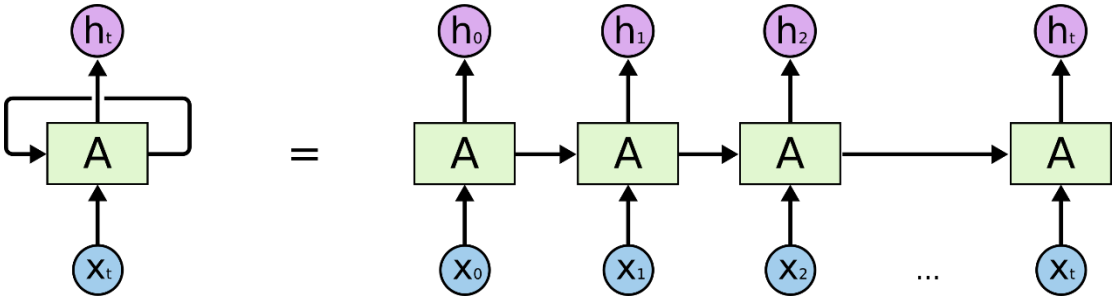

Figure 3-7 An unrolled RNN

An RNN is, simply put, a neural network with only one neuron that is run iteratively on loop. The unrolled RNN can be used to explain it a bit better.

Consider an input vector **X** of size *t,* and it is used to generate an output vector **h** of the same size. The first unit of **X, X_0** is passed through the RNN unit and using the random weights attached to it before, generates an output **h_0**. Then, when the second input is fed into the RNN, it not only used the fed input word, but it also uses the values of the previously trained word in tandem to generate **h_1.** This does on until the input vector is depleted and all the outputs are generated. Basically, by making use of the previous works' values, it considers the previous work as context to predict what the next value would be, by connecting the previous and the current word. Moreover, while these progresses, more words can be used to serve as context to predict the current word's value.

However, there is one glaring flaw in this system. It is that long term dependencies are generally erased. To combat this, LSTMs were made.

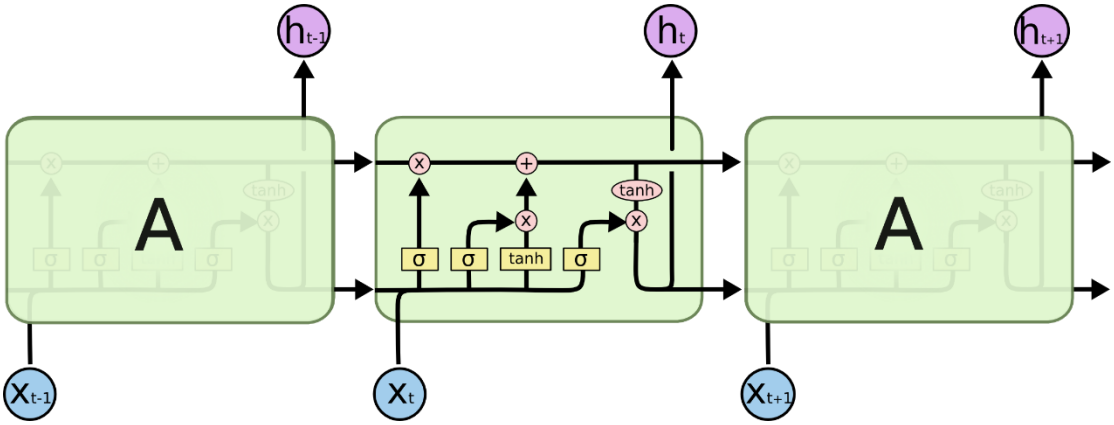

Figure 3-8 An LSTM unit

The unit described in figure 3-8 takes the place of **A** in figures 3-6 and 3-7. Instead of just one network layer, it has 4 layers (or gates).

The standout feature of the LSTMs is the line that passes at the top. This is what is used to carry the long-term dependencies throughout the network.

The way to calculate h_t is as follows:

$$f_t = \sigma(W_f.[h_t - 1, x_t] + b_f)$$

$$i_t = \sigma(W_i.[h_t - 1, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C.[h_t - 1, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

f_t, C_t, o_t, and h_t are the four most important values in this architecture.

The forget gate is used to determine whether the current value has to be forgotten or not within C. This is passed through a sigmoid function, that tells to either completely remove the value (0) or keep the value as it is (1).

The input gate layer is used to add the current input's info into C. First it is passed through a sigmoid and a tanh parallelly, then pointwise multiplied to get an update candidate.

The above two values (input and forget gates) are used to update the cell state C_{t-1} to a new cell state C_t.

Now, as for the output, we will pass the input through a sigmoid function and the cell state through a tanh activation. Then these values are pairwise multiplied to get h_t – which is the output of the current time step and also serves as a part of the input for the next time step. So, through the time steps, h will take care of all the immediate (short) context values and C will take care of the long-term dependencies, thus the name Long Short-Term Memory (or LSTM).

3.4.4 Various Classification Models

Since there are various classification models to choose from, we will perform a detailed analysis comparing the classification models to out problem as well as choose the best one from it. The models used are:

1. Logistic Regression
2. Gaussian Naïve Bayes
3. K-Neighbours Classifier
4. Decision Tree Classifier
5. Support Vector Classification
6. Random Forest Classifier
7. Shallow Neural Network Classifier

We did not make use of a deep neural network for the final classification as the amount of data available is extremely less to train such a model. Moreover, if we train it too much, it might overfit and regularization will not help, as that would lead to no learning since there is not enough data to mine classifiers from. We knew this because we tried. Anything more than one hidden layer overtrains even with Dropout and Batch Normalization.

4. DESIGN PROCESS

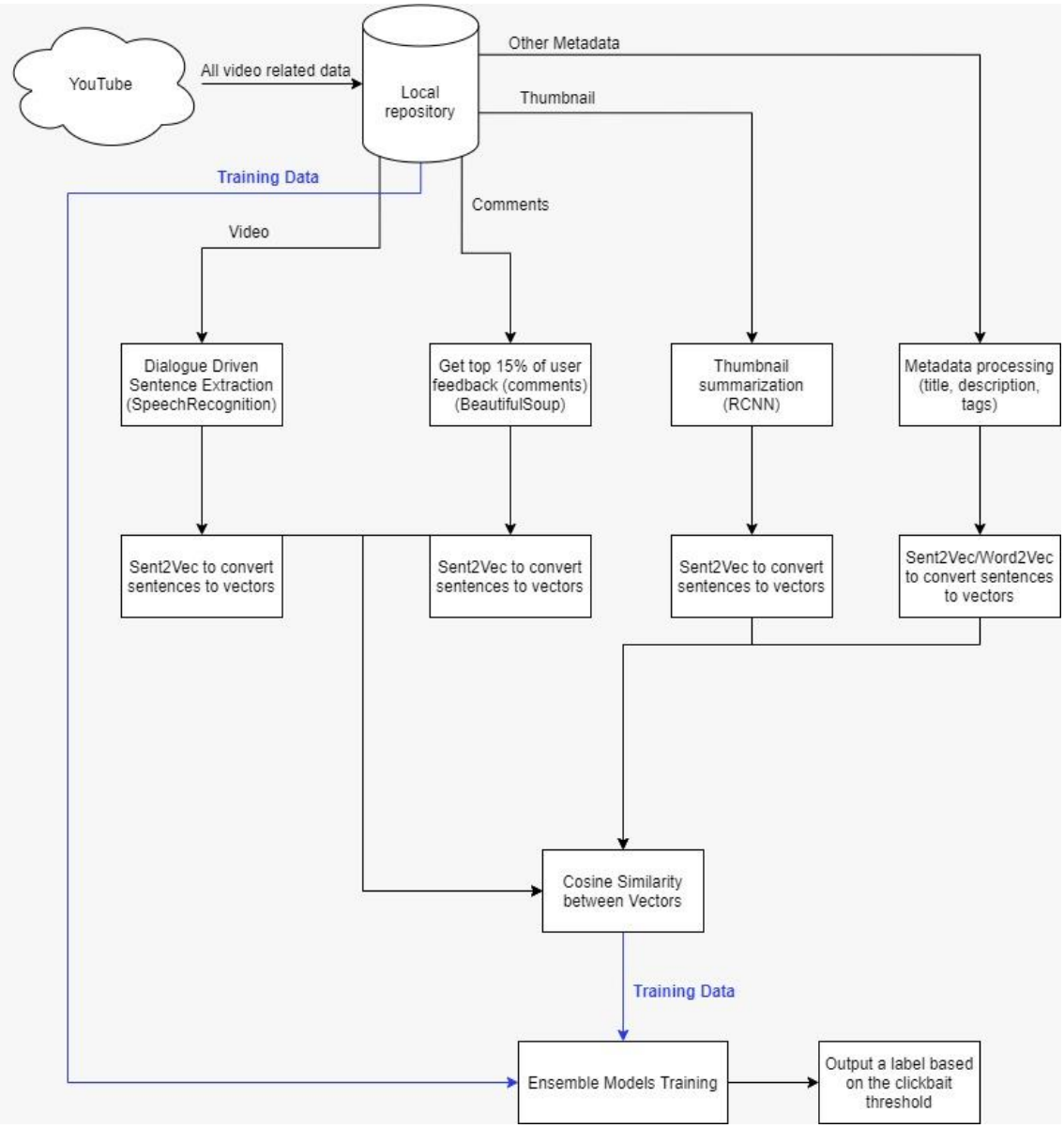

Figure 4-3-1 The project architecture

Figure 4-1 represents the project architecture in detail. We will take a look at the extraction of data during the implementation phase.

In essence, all the data concerning a video will be extracted from YouTube directly thanks to YouTube v3 API. We can perform a search over a broad term, say “life hacks”, and get the results back to us 50 videos at a time. Using the videoIDs that have been fetched, we can fetch the top 100 comments, thumbanil URL and the audio transcript of the video as well.

Thus, the list of features used can be surmised as in Table 4-1.

Table 4-1 Features used in our model and their sources

Values	Data Type	Location
Creator Fed Values		
Video_ID	String	YouTube API v3
Title	String	YouTube API v3
Tags	String	YouTube API v3
Description	String	YouTube API v3
Thumbnail Caption *(generated)*	String	RCNN Ensemble Model
Video Rating	String	YouTube API v3
Video Category ID	String	YouTube API v3
Uploaded At	Timestamp	YouTube API v3
Audio Transcript	String	Audio to Speech Recognition
User Interaction Values		
Likes	Integer	YouTube API v3
Dislikes	Integer	YouTube API v3
Like-Dislike Ratio	Float	Calculated
Number of comments	Integer	YouTube API v3
Number of "Fake" Comments	Integer	Calculated
"Fake" Comment Ratio	Float	Calculated
Views	Integer	YouTube API v3
Content Creator's Data		
Channel Age	Integer	YouTube API v3
Channel Total Views	Integer	YouTube API v3
Channel Total Subscribers	Float	Calculated
Channel Made for kids	Integer	YouTube API v3
Channel number of videos	Integer	Calculated

Then, data pre processing will take place. This includes processes like:

1. Converting categories into dummy one hot encodings.
2. Converting text into vectorized embeddings.
3. Replacing null and zero values with something more useful

a. For example, when comments were disabled, we were able to programmatically make a Google search of the video title to gather thoughts around the web instead of just dropping it entirely.

Then cosine similarity is found between the creator fed and user interaction values that were vectorized in order to reduce the dimensionality while still maintaining the relevancy of the features. These values along with the other numbered data are fed into the various classification models and the models are trained to classify clickbait from non-clickbait videos.

4.1 System Requirements

4.1.1 Hardware Requirements

S NO	PARAMETER	REQUIREMENT
1	CPU	2.4GHz 2-cores
2	GPU	12GB LPDDR5X VRAM
3	Memory	12GB DDR4 RAM
4	Disk Space	>= 25GiB
5	Monitor	As needed
6	Input Peripherals	As needed
7	Router	WiFi 5 and/or >= 15Mbps

Table 4-2 Hardware Requirements to run training and model

4.1.2 Software Requirements

S NO	PARAMETER	REQUIREMENT
1	OS	Windows 10 / MacOS Big Sur 11.3 / Ubuntu 18.04 or equivalent
2	Browser	Chromium-based Browser (or) Gecko driver-based Browser
3	Environment	Jupyter Notebook (or) Google Collaboratory
4	Coding Language	Python 3

Table 4-3 Software Requirements to run training and model

4.2 Module Specification

Here is a detailed list of all the python modules used:

4.2.1 Numpy

Numpy is a free python package used for various mathematical calculations and provides various data structure templates which are useful to perform various data manipulation operations with ease. Examples of such template are multidimensional array object, and derived objects like masked arrays and matrices. It also provides various mathematical, logical, shape manipulation, statistical operations and much more which are useful in data extraction and data manipulation

4.2.2 Pandas

Pandas is an open-source python package which provides a data frame object with integrated indexing. It is mainly used to load and manipulate the data that

exist as csv, excel or other dataset formats. The columns in pandas are mutable, which makes it easier to add and remove data as deemed by the user. It has various inbuilt functions like slicing, sub setting, reshaping, merging, joining etc., which helps the user manage the data more easily.

4.2.3 YouTube Transcript API

YouTube Transcript API is a 3rd Party API which allows the user to get the transcript, i.e., the caption provided for a YouTube video. If the uploader of the video has not included any captions, this package fetches the auto-generated caption provided by YouTube.

4.2.4 Beautiful Soup

Beautiful Soup is a python package which is used to scrape information from webpages. It is used for pulling data from HTML and XML files. It also includes various different parsers for different file formats.

4.2.5 Sentence Transformer

Sentence Transformer is python package that is used for text and sentence embedding. It is built based on Siamese BERT networks. The python package includes various pre-trained models with different metrics for different functionalities. This provides the user with the freedom using the required model for the user's specification and need.

4.2.6 Keras

Keras is deep learning API built on top of TensorFlow. It provides a simple and easy functions to help build complex neural network architectures. It acts an interface for the TensorFlow library. It contains various commonly used elements like layers, activation functions, optimizers etc., which are used to build neural networks.

4.2.7 TensorFlow

TensorFlow is a free and open-source library for training and loading machine learning models in python. It is based on dataflow and differential programming. It is used to import various pre-trained models and create different sessions for training and testing.

4.2.8 JSON

The python JSON package is used to encode and decode json files. It is used to store dictionaries as json files for later access.

4.2.9 Scikit- Learn

Scikit is learn is python package which provides various machine learning models for training and various other packages that assist in training of the model. It provides modules which are used to split the training and testing data, used to calculate the accuracy of the model, determine the precision, recall, F-score etc.

4.2.10 Random Forest Classifier

This machine learning model is built based on the Decision-Tree classifier. The model consists of a set of Decision trees derived from the test data and provides the classification based on the collection of results of all the decision trees.

4.2.11 Logistic Regression

This model is a classification model that uses logistic functions for binary classification. The most commonly used function for binary classification is the sigmoid function

4.2.12 K-Neighbors Classifier

K-Neighbors classifier is a supervised machine learning algorithm which classifies the data based on the K nearest neighbors of the data. The nearest neighbors are identified based of the distance between the input data and the data used for training. The distance most commonly used is the Euclidean Distance.

4.2.13 Support Vector Classifier

Support Vector Classifier is a supervised classification algorithm where the input data are vectors and the algorithm focuses on creating a hyperplane which separates the data into different classes.

4.2.14 Gaussian Naïve Bayes

The Gaussian Naïve Bayes classifier is supervised machine learning algorithm which classifies the data based on the likelihood of a “event” occurring, i.e., the likelihood that the given data is classified as Class 1 or 2 based on the values of the feature provided as input.

5. IMPLEMENTATION

5.1 Data Extraction

The dataset is obtained using the 2 different APIs

- YouTube Data API
- YouTube Transcript API

The YouTube Data API is the official API provided by Google to get all video-related information. The YouTube Transcript API is a legal 3rd party API that provides us with the transcript of a YouTube video. The reason for opting for a third-party API is that it is illegal to download the video file or the audio file of a YouTube video, which is required for us to process the video using any of the Speech Recognition models.

5.2 Data Pre – Processing

We were able to successfully retrieve 570 videos and metadata concerning each video. Now, moving onto data pre-processing, we used SBERT. It is the sentence vectorizer used to vectorize all the words and sentences in the dataset. SBERT is a python framework for sentence and text embedding. This sentence and text embedder is designed using Siamese-BERT networks. This package included various pre-trained models along with different parameters of each model to help choose the model for the user's requirement. The model chosen by us to embed the text is called 'paraphrase-multilingual-mpnet-base-v2', which had good performance across all metric like STS Benchmarks, speed, and also had the highest accuracy in embedding Twitter paraphrases among all the pre-trained models available. The domain we are working on is also social media, this was the ideal model for us to vectorize the text data in the dataset.

The search term is given as the input to a function that returns a list of unique VideoID of videos. Using the list obtained, we obtain all video-related information like the title of the video, description, the number of likes, dislikes and views it has obtained, the category to which the video belongs, the tags associated with the video. After obtaining all video-related information, using the Channel ID, the ID of the channel in which the video is uploaded, the

YouTube channel-related information like the number of subscribers to that channel, and the number of videos uploaded by the channel is collected. Using the VideoID, the transcript is obtained by getting the autogenerated caption provided by YouTube for the video.

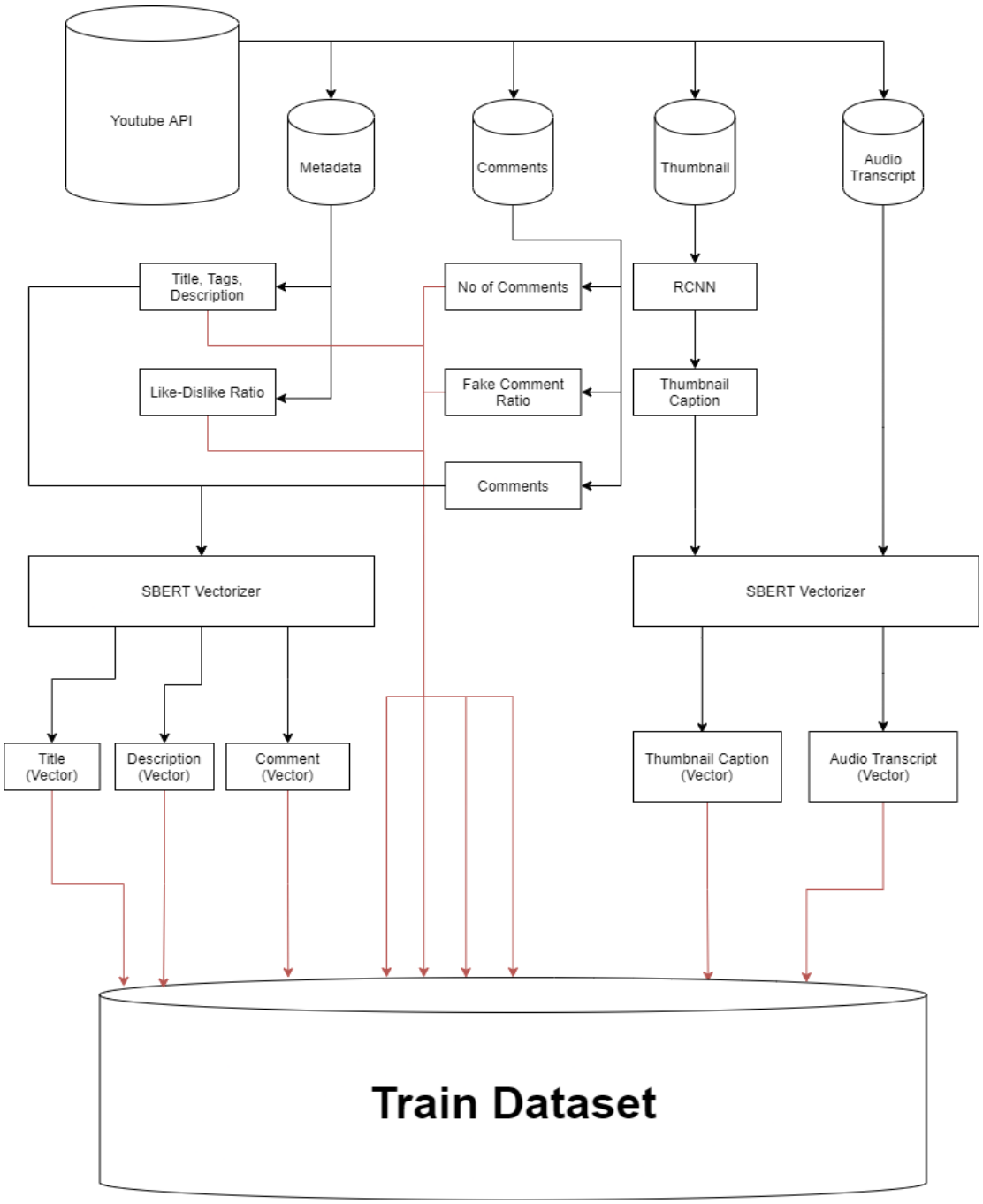

Figure 5-1 Dataset preparation pipeline

The hyperlink to the image in the video's thumbnail can also be obtained using the VideoID.

Comments and comment threads are also collected using the same VideoID. If the comments of a certain video are disabled, a google search is executed where the search query is the title of the video along with the word ‘fake’ added to it. The titles of the top search results are used in place of the comments.

From the number of likes and dislikes, the like-dislike ratio is calculated for each video. This gives us an idea of how much the given video is liked or disliked by users. The comments are extracted and preprocessed using the NLP methods like tokenization and keyword extraction. A set of unique words which are associated with clickbait is created. The intersection between the set of keywords from the comments and the created set of clickbait words helps us determine the no. of comments which claim the video to be fake or clickbait. This when combined with the total no of comments gives us the ratio of the comments which claim the video to be clickbait among all the comments of the video. The image of the thumbnail is downloaded using the link obtained earlier and using an RCNN, the caption of the image is generated.

All the words and sentences in the dataset are vectorized using a sentence transformer called SBERT. After all the vectors are generated, the cosine similarity is calculated among the vectors of the user-generated data and the vectors of the uploaded generated data. The cosine similarity is calculated to find out how different or how similar the content of the video is to the title and thumbnail of the video since the title and thumbnail are the reason for a user to click on a video.

When it comes to sentence embedding, note that the words are positionally encoded, otherwise, the model would just look at the sentences like a typical bag of words.

SBERT improves upon BERT and Transformers by imploying two parallel BERTs on the two sentences to be compared, pools them (average or max pooling) and computes the softmax to classify the sentences as similar or dissimilar. It is worthy to note that the two BERTs have tied weights, hence the name Siamese model. Although average GLoVe embeddings are still the fastest method to compute sentence embeddings, they are not as accurate in maintaining the relationship between words like SBERT, thus we chose SBERT

as context and word choice of titles, the deliberation behind making thumbnails, and the feedback of the users in comments are all extremely important.

Next up, we make use of an RCNN ensemble model to generate a caption for the thumbnail and create a sentence embedding of it using SBERT.

RCNNs are extensively used in the field of computer vision, especially related to videos, and here it is no different. We make use of the Inception V3 pre-trained model to generate an embedding of the thumbnail first and then, feed it into a custom LSTM to decode the embeddings in order to generate a caption for the thumbnail, which is in line with the Show and Tell model discussed in [4].

Inception V3 is an extremely deep convolutional neural network developed by Google that is used for object recognition in images and classification of images. It famously won the runner in the ImageNet classifier challenge. The reason we didn't choose Microsoft's Resnet was simply because it is just too computationally heavy and thus, infeasible in our setup at least. Regardless, ResNet (3.6%) is better than InceptionV3 (6.6%) by only 3.1% and is also below the Bayesian limit for image recognition tasks (which is an error rate of about 5.1% for humans). There is no sense in choosing a model that performs better than humans since clickbait is used to entice users, and if the enticing factor of the video is not even recognizable to the humans, then the bait is technically not doing a good job at baiting.

We had to make a custom LSTM model for our use case since there aren't any available that is open source and suit our requirements. We make use of 300 LSTM units and we bottleneck the logits and the image embedding to 120 to reduce the number of trainable parameters.

After painstakingly going through each video and considering other parameters pertaining to them using the data gathered from above, we were able to label our dataset as follows:

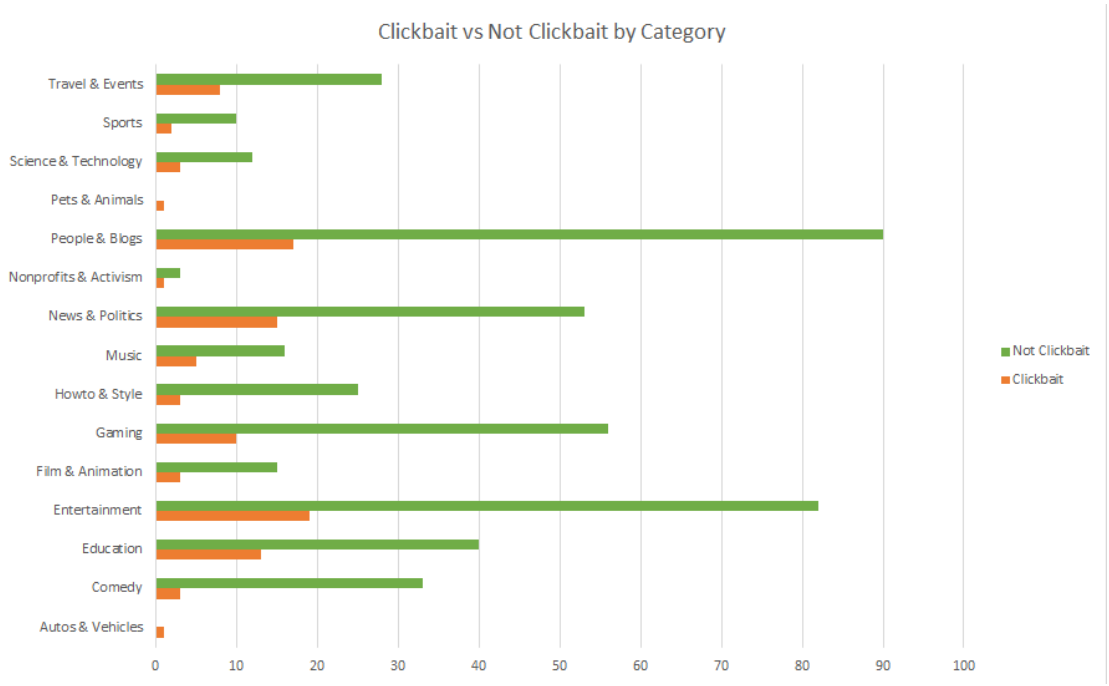

Figure 5-2 Category-wise distribution of clickbait and non-clickbait videos

Finally, in order to determine the best possible classifier for our model we have made a simple baseline if-else based classifier, a logistic regression model, a support vector classifier, a random forest implementation, a decision tree implementation and a gaussian Naïve Bayes implementation. We will draw a comparison of all of the above by making use of various metrics like recall, precision, F1-score accuracy and ROC-AUC graphs.

6. CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

We were able to train our model on the aforementioned seven classifiers. Unfortunately, even a shallow neural network proved to be impractical given the amount of data and features that we possess for our project. Nevertheless, the remaining six models were able to run, with varying degrees of success.

Of the six, Gaussian Naïve Bayes proved to be the worst, and the reason might be because it is primarily used for continuous values and we had a categorical feature in our dataset.

Classification Reports

1. Logistic Regression

	0	1	accuracy	macro avg	weighted avg
precision	0.898734	0.571429	0.872093	0.735081	0.853064
recall	0.959459	0.333333	0.872093	0.646396	0.872093
f1-score	0.928105	0.421053	0.872093	0.674579	0.857353
support	74	12	0.872093	86	86

2. Gaussian Naive Bayes

	0	1	accuracy	macro avg	weighted avg
precision	0.947368	0.164179	0.337209	0.555774	0.838086
recall	0.243243	0.916667	0.337209	0.579955	0.337209
f1-score	0.387097	0.278481	0.337209	0.332789	0.371941
support	74	12	0.337209	86	86

3. K-Nearest Neighbours

	0	1	accuracy	macro avg	weighted avg
precision	0.935065	0.777778	0.918605	0.856421	0.913118
recall	0.972973	0.583333	0.918605	0.778153	0.918605
f1-score	0.953642	0.666667	0.918605	0.810155	0.913599
support	74	12	0.918605	86	86

Table 6-1 Classification Reports - 1

4. Decision Tree Classifier

	0	1	accuracy	macro avg	weighted avg
precision	0.958904	0.692308	0.918605	0.825606	0.921705
recall	0.945946	0.75	0.918605	0.847973	0.918605
f1-score	0.952381	0.72	0.918605	0.83619	0.919956
support	74	12	0.918605	86	86

5. Support Vector Classifier

	0	1	accuracy	macro avg	weighted avg
precision	0.9125	0.833333	0.906977	0.872917	0.901453
recall	0.986486	0.416667	0.906977	0.701577	0.906977
f1-score	0.948052	0.555556	0.906977	0.751804	0.893285
support	74	12	0.906977	86	86

6. Random Forest Classifier

	0	1	accuracy	macro avg	weighted avg
precision	0.960526	0.9	0.953488	0.930263	0.952081
recall	0.986486	0.75	0.953488	0.868243	0.953488
f1-score	0.973333	0.818182	0.953488	0.895758	0.951684
support	74	12	0.953488	86	86

Table 6-2 Classification Reports – 2

Regardless, an accuracy of only **33.72%** is worse than guessing blindly. Thus, it became obvious that we had to move on.

The best classifiers from our testing have to be the tree-based classifiers i.e., Decision Tree classifier, KNN Classifier, and the winner, Random Forest Classifier. Coming in with an accuracy of **95.35%**, it was the best performing model by over 3.49%! Moreover, it's precision and recall were not bad as well, compared to other models at least. The only negative, at least in terms of metrics is that, its recall when it comes to classifying positive labels was only 75%, but that is the case because the distribution of positive and negative labels were off.

In order to understand the disparity in performance, let us take a look at the ROC-AUC graph as well.

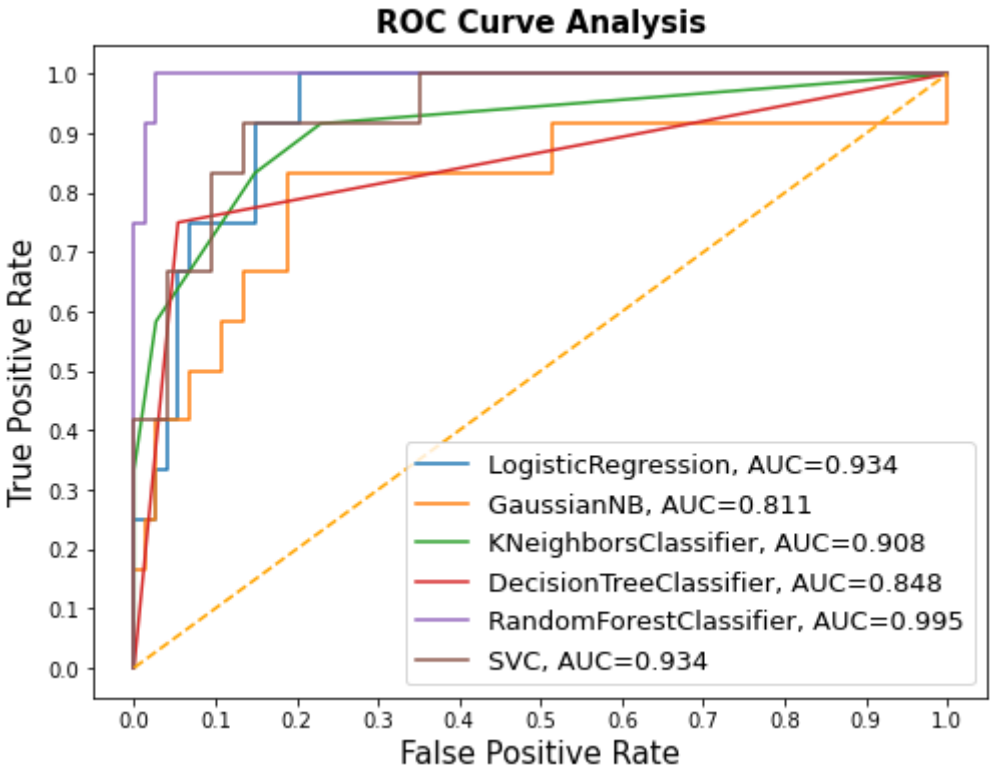

Figure 6-1 ROC-AUC values of different models

Clearly Random Forest outperforms the others. Therefore, the classifier in our final model will be the Random Forest Classifier.

6.2 Future Enhancements

Although we were able to make a YouTube video clickbait classifier with decent accuracy, this venture is far from over. Clickbait is an extremely subjective topic and each person views clickbait differently. Some may say, as long as they get the subject matter of the title and thumbnail in the video, it should not be clickbait, regardless of the length of the video. On the other hand, others might retort by saying that if the video is advertised as a specific subject, the entirety of the video should be about said subject matter. It is a never-ending battle where no side will get satisfied with the others' views and opinions.

Therefore, we have ample improvements in mind. One of which includes actually considering the video's action-driven content that was explained earlier. The models exist, but training in the current infrastructure that we have in place is unfeasible to say the least. So, a person with sufficient amount of

time and computing resources can definitely take advantage of S2VT and improve this classifier even further.

Another improvement would be to actually improve the user interaction-based values. One idea that we had in mind was to develop an extension that predicts whether the YouTube video that is currently playing is clickbait or not, then ask for feedback from the user in the form of a like or dislike. Based on the amount likes and dislikes per prediction, we can periodically retrain the model in order to achieve much better results that satisfies a larger group of viewers.

One final, albeit far-fetched enhancement would be for YouTube to adopt this model and implement it within YouTube Creator and analytics that are being used by YouTubers. This model can be used to rely solely on the video content and the creator fed values and notify the uploader whether the video is clickbait or not, and if required, change the title and thumbnail of the video before uploading. YouTube has more than enough resources and talent in its hand to implement such a model, but we doubt it would ever come to fruition because clickbait is what earns YouTube revenue as well. YouTube takes 45% of the advertising revenue generated per video, so it would be a dumb business move YouTube's part to implement this. Nevertheless, we can stay optimistic.

That being said, it is our responsibility as viewers to learn and identify clickbait in the videos that we watch and, subsequently prevent sharing and spreading them in order to have a more truthful and informative future in media.

7. REFERENCES

[1] Sarjak Chawda, Aditi Patil, Abhishek Singh, Prof. Ashwini Save **,' A Novel Approach for Clickbait Detection'** - Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019) IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-8

[2] E. Uzun**, "A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages,"** in IEEE Access, vol. 8, pp. 61726-61740, 2020, doi: 10.1109/ACCESS.2020.2984503.

[3] Meng Wang, Richang Hong, Guangda Li, Zheng-Jun Zha, Shuicheng Yan, Tat-Seng Chua **'Event Driven Web Video Summarization by Tag Localization and Key-Shot Identification'** in IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 14, NO. 4, AUGUST 2012

[4] Andrej Karpathy,Li Fei-Fei **'Deep Visual-Semantic Alignments for Generating Image Descriptions' –** Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3128-3137

[5] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama,Kate Saenko, Trevor Darrell '**Long-term Recurrent Convolutional Networks for Visual Recognition and Description'**

[6] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan **'Show and Tell: A Neural Image Caption Generator' -** Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3156-3164

[7] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, Yoshua Bengio **'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention' -** Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:2048-2057, 2015.

[8] Subhashini Venugopalan,Marcus Rohrbach,Jeff Donahue,Raymond Mooney ,Trevor Darrell,Kate Saenko **"Sequence to Sequence – Video**

to Text" - Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4534-4542

[9] Abinash Pujahari and Dilip Singh Sisodia **"Clickbait Detection using Multiple Categorization Techniques"**

[10] Shu, Kai & Wang, Suhang & Lee, Dongwon & Liu, Huan. **"(2020). Mining Disinformation and Fake News: Concepts, Methods, and Recent Advancements"** 10.1007/978-3-030-42699-6_1.

[11] Behrooz Mahasseni, Xiaodong Yang, Pavlo Molchanov, & Jan Kautz. (2018). "**Budget-Aware Activity Detection with A Recurrent Policy Network."**

APPENDIX

A1. Source Code

A1.1 Data Extraction and Pre-processing:

```
# Import all necessary packages

import requests

import os

import bs4

from apiclient.discovery import build

import csv

import datetime

import string

from youtube_transcript_api import YouTubeTranscriptApi

import csv

import json

import pandas as pd

from sentence_transformers import SentenceTransformer

# Loading the pre-trained model for text embedding

vectorize = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')

# Parameters for YouTube API

DEVELOPER_KEY = "AIzaSyDqlrLX0prKppP3eG02uqprQJk080XoKrc"

YOUTUBE_API_SERVICE_NAME = "youtube"

YOUTUBE_API_VERSION = "v3"

youtube_object = build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION,developerKey = DEVELOPER_KEY)

# Main Dictionaries to store the retrieved values
```

```
MetaDataMain = {}
CommentsDataMain = {}
next_page_token = ""
# Function to get the video related data
def getdata(query, max_results, pageToken=""):
    meta_data = {}
    comments_data = {}
    # Get a list of Video IDs
    search_keyword = youtube_object.search().list(q = query, type = "video",
                                        part = "id, snippet",  order = 'viewCount',
                                        maxResults = max_results, pageToken =
pageToken).execute()
    nextPageToken = search_keyword["nextPageToken"]
    results = search_keyword.get("items", [])
    v_id = []
    for result in results:
        if result['id']['kind'] == "youtube# video":
            v_id.append(result["id"]["videoId"])
    flag = {}
    for i in v_id:
        flag[i] = True
        meta_data[i] = {}
    # Get Audio Transcript
    for i in v_id:
        try:
```

```
            transcript = YouTubeTranscriptApi.get_transcript(i)
            meta_data[i]["audioTranscript"] = [" ".join(t['text'] for t in transcript)]
        except Exception as e:
            meta_data[i]["audioTranscript"] = "No Transcript Found"
    # Get meta-data of videos
    for i in v_id:
        print("Processing V-Id", i)
        if flag[i] == False:
            continue
        try:
            temp = {'videoId': i}
video_stat=youtube_object.videos().list(part='snippet,statistics,contentDetails',id=i).execute()
channel_stat=youtube_object.channels().list(part='status,snippet,statistics',id=video_stat['items'][0]['snippet']['channelId']).execute()
category_stat=youtube_object.videoCategories().list(part='snippet',id=video_stat['items'][0]['snippet']['categoryId']).execute()
            temp['title'] = video_stat['items'][0]['snippet']['title']
            temp['likes'] = video_stat['items'][0]['statistics']['likeCount'] if 'likeCount' in video_stat['items'][0]['statistics'] else -1
            temp['views'] = video_stat['items'][0]['statistics']['viewCount']
            temp['dislikes'] = video_stat['items'][0]['statistics']['dislikeCount'] if 'dislikeCount' in video_stat['items'][0]['statistics'] else -1
            temp['thumbnail'] = video_stat['items'][0]['snippet']['thumbnails']['medium']['url']
            temp['category'] = category_stat['items'][0]['snippet']['title']
```

```
            temp['ldRatio'] = (int(temp['likes']) - int(temp['dislikes'])) /
(int(temp['likes']) + int(temp['dislikes'])) if 'likes' in temp and 'dislikes' in temp
else None

            temp['channelVideos'] =
channel_stat['items'][0]['statistics']['videoCount'] if 'videoCount' in
channel_stat['items'][0]['statistics'] else -1

            temp['channelSubscribers'] =
channel_stat['items'][0]['statistics']['subscriberCount'] if 'subscriberCount' in
channel_stat['items'][0]['statistics'] else -1

            temp['channelMadeForKids'] =
channel_stat['items'][0]['status']['madeForKids'] if 'madeForKids' in
channel_stat['items'][0]['status'] else False

            meta_data[i].update(temp)

        except Exception as e:

            print("Some error with keys, exit")

            print(e)

            continue

    # Get comments of the video

    for i in v_id:

        if flag[i] == False:

            continue

        temp = []

        try:

            try:

video_response=youtube_object.commentThreads().list(part='snippet,replies',
videoId=i,maxResults=100).execute()

                for item in video_response['items']:
```

```
                comment =
item['snippet']['topLevelComment']['snippet']['textDisplay']
                temp.append(comment)
            comments_data[i]= temp
        except:
            print("Disabled Comments, get search URLs")
            url = 'https://google.com/search?q='
            request_result=requests.get( url + meta_data[i]['title'] + " fake")
            soup = bs4.BeautifulSoup(request_result.text,
                                     "html.parser")
            temp = [i.getText() for i in soup.find_all( 'h3' )]
            comments_data[i]= temp
            pass
    except Exception as e:
        print("Disabled Comments and error with urls, exit")
        print(e)
        continue
  return meta_data, comments_data, nextPageToken
# The data-fetch function is called repetetively until the required number of
vidoe-related data is obtained.
while len(MetaDataMain) < 175:
  meta_data, comments_data, next_page_token =
getdata('tiktok',25,next_page_token)
  MetaDataMain.update(meta_data)
  CommentsDataMain.update(comments_data)
```

```
# Calculate the number of comments which claim the video is fake
clickbait_words_corpus = {"fake", "clickbait", "click", "bait", "lie", "false",
                          "waste", "thumbnail", "title", "subscribe", "subscribed",
                          "wasted", "useless", "bullshit", "baited", "rat",
                          "misinformation", "credible", "crap", "tosh", "dumb",
                          "life", "time", "unreliable", "lack", "bogus", "fraud",
                          "phony", "scam", "sham", "trick",
"cheat","hoax","rumour","rumor",
                          "gossip"}
fake_comment_data = {}
temp = MetaDataMain
for v_id in CommentsDataMain:
    count_fake_comment = 0
    for comment in CommentsDataMain[v_id]:
        processed_comment = comment.translate(str.maketrans('', '',
string.punctuation))
        unique_words_comment = set(processed_comment.split(" "))
        if len(clickbait_words_corpus.intersection(unique_words_comment)) >=
1:
            count_fake_comment += 1
    try:
        fake_comment_ratio = count_fake_comment /
len(CommentsDataMain[v_id])
    except ZeroDivisionError:
        fake_comment_ratio = -1
```

```
    temp[v_id].update({"noOfComments":
len(CommentsDataMain[v_id]),"noOfFakeComments":
count_fake_comment,"fakeCommentRatio": fake_comment_ratio})

MetaDataMain = temp

# Initialize the list of columns to vectorize

vecList = ['title','description','thumbnailCaption','commentsData']

# Vectorizing the columns

for i in vecList:

  df[i] = df[i].apply(lambda x: vectorize.encode(x))

# Drop the unnecssary columns

df = df.drop(columns=['videoId','thumbnail','Unnamed: 0'])

# Initialize the list of columns for the which the values need to be normalized
before training

normlist =
['likes','dislikes','views','publishedAt','channelAge','channelViews','channelSub
scribers','channelVideos']

# Normalizing the values

from sklearn import preprocessing

for i in normlist:

  x = df[[i]].values.astype(float)

  min_max_scaler = preprocessing.MinMaxScaler()

  x_scaled = min_max_scaler.fit_transform(x)

  df[i] = x_scaled

# Saving the processed data as a new CSV file

df.to_csv("VecDataSetCategorized.csv")
```

A1.2 Model Training and Evaluation

```
# Import all necessary Packages

import pandas as pd

import numpy as np

import seaborn as sb

from sklearn.model_selection import train_test_split

# Loading the dataset to a dataframe

tempdf = pd.read_csv("FinalTrainDataset2.csv")

tempdf.head()

# Separating the input features and output label into different variables for
training

x = finaldf.iloc[:,:-1]

y = finaldf.iloc[:,-1]

# Splitting the data for training and testing

X_train , X_test, y_train, y_test = train_test_split(x,y,test_size = 0.15)

len(X_train),len(X_test)

# Import all necessary packages

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import roc_curve, roc_auc_score
```

```
import sklearn
# Initializing the classifiers for training, in a list
classifiers = [LogisticRegression(random_state=1234),
               GaussianNB(),
               KNeighborsClassifier(),
               DecisionTreeClassifier(random_state=1234),
               RandomForestClassifier(random_state=1234),
               SVC(probability=True)]
#  Creating a dataframe to draw charts based on the results
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])
for cls in classifiers: #  Training all the classifiers in the list
    model = cls.fit(X_train, y_train)
    yproba = model.predict_proba(X_test)[::,1]
    y_pred = model.predict(X_test)
    fpr, tpr, _ = roc_curve(y_test,  yproba)
    auc = roc_auc_score(y_test, yproba)
    result_table = result_table.append({'classifiers':cls.__class__.__name__,
                                        'fpr':fpr,
                                        'tpr':tpr,
                                        'auc':auc}, ignore_index=True)
result_table.set_index('classifiers', inplace=True)
# Plotting the metrics of the models in a graph
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
```

```
for i in result_table.index:
    plt.plot(result_table.loc[i]['fpr'],
             result_table.loc[i]['tpr'],
             label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))
plt.plot([0,1], [0,1], color='orange', linestyle='--')
plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("False Positive Rate", fontsize=15)
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("True Positive Rate", fontsize=15)
plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15)
plt.legend(prop={'size':13}, loc='lower right')
plt.show()
```

A1.3 Model in Action

```
# Import all necessary packages
import requests
import os
import bs4
from apiclient.discovery import build
import string
from youtube_transcript_api import YouTubeTranscriptApi
import csv
import json
import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn import preprocessing
```

```
from textblob import TextBlob
from scipy.spatial import distance
import pickle
# Load the text embedding model
vectorize = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')
# Initialize the parameters for the Youtube API
DEVELOPER_KEY = "AIzaSyDqlrLX0prKppP3eG02uqprQJk080XoKrc"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
# Build the Youtube API Object
youtube_object = build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION,developerKey = DEVELOPER_KEY)
# Function to classify a video as Clickbait or Not Clickbait
# vID is the Video ID of the Youtube Video
def isclickbait(vID):
  # Loading the model
  model = pickle.load(open('/content/isclickbait.sav','rb'))
  # Creating the object used for normalizing the values
  mms = preprocessing.MinMaxScaler()
  # Words often associated with clickbait stored as set to find intersection
  clickbait_words_corpus = {"fake", "clickbait", "click", "bait", "lie", "false",
                            "waste", "thumbnail", "title", "subscribe", "subscribed",
                            "wasted", "useless", "bullshit", "baited", "rat",
                            "misinformation", "credible", "crap", "tosh", "dumb",
                            "life", "time", "unreliable", "lack", "bogus", "fraud",
```

```
                    "phony", "scam", "sham", "trick",
"cheat","hoax","rumour","rumor",
                    "gossip"}
  # Range of data of the features used in training, required for normalizing the
data
  ranges = {'likes':[[-1],[10051603]],
          'views' :  [[367],[213578965]],
          'dislikes': [[-1],[386489]],
          'noOfComments' : [[0],[100]],
          'count_fake_comment' : [[0],[20]],
          'neg_comments' : [[0],[37]]
          }
  # Extract video related data / meta-data

video_stat=youtube_object.videos().list(part='snippet,statistics,contentDetails',
id=vID).execute()
  likes = video_stat['items'][0]['statistics']['likeCount'] if 'likeCount' in
video_stat['items'][0]['statistics'] else -1
  views = video_stat['items'][0]['statistics']['viewCount']
  dislikes = video_stat['items'][0]['statistics']['dislikeCount']
  ldRatio = (int(likes) - int(dislikes))/(int(likes) + int(dislikes))
  # Extract the comments of the video
  comment_data = []
  try:
video_response=youtube_object.commentThreads().list(part='snippet,replies',
videoId=vID,maxResults=100).execute()
```

```
    for item in video_response['items']:
                comment =
item['snippet']['topLevelComment']['snippet']['textDisplay']
                comment_data.append(comment)
  except:
    print("Disabled Comments, get search URLs")
    url = 'https://google.com/search?q='
    request_result=requests.get( url + meta_data[i]['title'] + " fake")
    soup = bs4.BeautifulSoup(request_result.text,
                    "html.parser")
    comment_data = [i.getText() for i in soup.find_all( 'h3' )]



  # Extract the audio transcript of the video
  try:
    audioTranscript = YouTubeTranscriptApi.get_transcript(vID)
    transcript = ""
    for i in audioTranscript:
      transcript += i['text'] + " "
  except Exception as e:
      transcript = " ".join(comment_data)
  noOfComments = len(comment_data)
  # Count the no of comments which claim the video to be 'fake' or 'Clickbait'
  count_fake_comment = 0
  neg_comments = 0
```

```
  for comment in comment_data:
      processed_comment = comment.translate(str.maketrans('', '',
string.punctuation))
      if TextBlob(comment).sentiment.polarity < 0:
        neg_comments += 1
      unique_words_comment = set(processed_comment.split(" "))
      if len(clickbait_words_corpus.intersection(unique_words_comment)) >= 1:
          count_fake_comment += 1
  try:
      # Calculate Fake Comment Ratio and Positive comment ratio
      fake_comment_ratio = count_fake_comment / len(comment_data)
      pos_ratio = (noOfComments - neg_comments) / len(comment_data)
  except ZeroDivisionError:
      fake_comment_ratio = -1
      pos_ratio = -1
  # Get the title of the video
  title = video_stat['items'][0]['snippet']['title']
  # Vectorize the title of the video
  title = vectorize.encode(title)
  # Vectorize the audio transcript of the video
  transcript = vectorize.encode(transcript)
  # Find the cosine similarity of the above vectors
  titleVtranscript = distance.cosine(title,transcript)
  # Normalizing the data using the initialized ranges
  ranges['likes'].insert(1,[likes])
```

```
  ranges['views'].insert(1,[views])
  ranges['dislikes'].insert(1,[dislikes])
  ranges['noOfComments'].insert(1,[noOfComments])
  ranges['count_fake_comment'].insert(1,[count_fake_comment])
  ranges['neg_comments'].insert(1,[neg_comments])
  likes = mms.fit_transform(ranges['likes'])[1][0]
  views = mms.fit_transform(ranges['views'])[1][0]
  dislikes = mms.fit_transform(ranges['dislikes'])[1][0]
  noOfComments = mms.fit_transform(ranges['noOfComments'])[1][0]
  count_fake_comment =
mms.fit_transform(ranges['count_fake_comment'])[1][0]
  neg_comments = mms.fit_transform(ranges['neg_comments'])[1][0]
  # Storing all the data collected in a single list for prediction
  data =
[likes,views,dislikes,ldRatio,noOfComments,count_fake_comment,fake_comm
ent_ratio,titleVtranscript,neg_comments,pos_ratio]
  # Prediction
  res = (model.predict_proba([data])[::,1]*100)[0]
  # Print the results
  print("Our Model predicted that this video is {:.2f}% clickbait".format(res))
```

A2. Screenshots

a baseball player is swinging a bat at a baseball

Figure A2-1 A sample caption generated for a thumbnail

```
isclickbait('ZN82P5RUW6E')
```

```
Our Model is 75.64% confident that this video is clickbait
```

Figure A2-2 A clickbait video as predicted by our model

```
isclickbait('xDzcHagoYqw')
```

```
Our Model is 10.12% confident that this video is clickbait
```

Figure A2-3 A non-clickbait video as predicted by our model

TECHNICAL BIOGRAPHY

VISHAL R K

Vishal R.K (170071601143) was born on 5th December,1999 Chennai, Tamil Nadu. He completed 10th Grade in Srimathi Sundaravalli Memorial School CBSE with a CGPA of 10. He completed his 12th Grade in Srimathi Sundaravalli Memorial School CBSE with the aggregate score of 95.6%. He is currently pursuing his B.Tech in Computer Science and Engineering from B.S. Abdur Rahman Crescent Institute of Science and Technology. His area of interest includes Natural Language Processing, Social-Media. Deep Neural-Networks, Generative Networks

E-Mail: vishalk99rkv@gmail.com

VISHAL KRISHNAN S H

Vishal Krishnan S.H (170071601142) was born on 29th September, 1999 Chennai, Tamil Nadu. He completed 10th Grade in Vels Vidyashram with a CGPA of 8.6. He completed his 12th Grade in Kendriya Vidyalaya, I.I.T Chennai with the aggregate score of 82.4%.He is currently pursuing her B.Tech in Computer Science and Engineering from B.S. Abdur Rahman Crescent Institute of Science and Technology. His area of interest includes Computer Vision and Machine Learning.

E-Mail: vishalk2999@gmail.com