Fine-Tuned Large Language Models for Improved Clickbait Title Detection

Chandra N Sekharan, Ph.D
Department of Computer Science
Texas A&M University Corpus Christi
Corpus Christi, TX 78412
csekharan@tamucc.edu

Pavan Sai Vuppala
Graduate Student
Department of Computer Science
Texas A&M University Corpus Christi
Corpus Christi, TX 78412
pvuppala1@islander.tamucc.edu

***Abstract*—The term *clickbait* is used to describe headlines or other content that is designed to attract attention and clicks, often by using sensationalized or misleading language. Clickbaiting is a significant problem for online users, as it often leads to low-quality content negatively impacting user engagement and trust. However, the task of detecting click-bait titles automatically can be challenging due to the wide variance in the wording and structure of titles. Manually detecting them can also be problematic because different people have different opinions on what constitutes click-bait. Rule-based approaches, machine learning models, deep learning models, and natural language processing techniques are some of the existing methods for click-bait detection. As good as these techniques are, they still need customization and large amounts of training data to achieve good accuracy. Since generative, pre-trained transformer models have shown an inherent capability to understand the meanings of sentences really well, our research examined the use of GPT-3.5, and GPT-4 as zero-shot models. They resulted in a maximum test accuracy of 88.5%, lower than some of the custom techniques reported in previous research works of between 90% to 95%. This indicates the need to go beyond a generic LLM. We then fine-tune OpenAI's Large Language Model (LLM) Ada using an efficient dataset of only 1000 samples of clickbait and non-clickbait titles to get a test accuracy of up to 99.5 %. The research further shows that fine-tuning a GPT model is not only more accurate but also uses a smaller amount of training data, minimal coding, and cost-effective. Our research is indicative of the broader possibility that fine-tuning a large language model is all you need for most targeted natural language processing tasks. All our research experiments used OpenAI APIs using Python notebooks on Google Colab and the data and code are publicly available in a GitHub repository.**

Keywords: Machine learning, OpenAI API, Click-Bait Detection, LLM, Fine-tuning, Ada, GPT-4

I. Introduction

There has been an exponential rise in the use of clickbait, a type of sensationalized title designed to attract users' attention and persuade them to click on an associated URL. Clickbait titles such as *Which Zodiac Sign Should You Date Based On Your Favourite Disney Character* or *This Quiz Will Tell You Which Breakfast Cereal Matches Your Personality* have become increasingly common on news and social media platforms. These types of titles, unlike non-clickbait titles such as *Arctic ice thickness decreasing, suggests satellite data study* or *Basketball: Lakers score 102 to defeat the Celtics in Game 1 of the 2010 NBA Finals*, do not accurately describe the content on the linked page. This can result in user dissatisfaction and potentially dangerous consequences if an attacker infuses malware with the click. Various methods have been proposed to identify clickbaits, including using supervised machine learning techniques such as SVM, random forest models, and long-short term memory. In this research paper, we propose a refined approach to detecting clickbait using openAI's Ada, a Large Language model based on the GPT-3 architecture.

Machine learning has made remarkable strides in recent years owing to its impact in various areas today. Supervised and unsupervised learning are two of the most significant techniques used in machine learning. With supervised learning, the machine is trained on labeled data, which means that each data point has a label associated with it that helps define the data. After being trained on this labeled data, the machine or model is tested to see if it can accurately produce the desired outcomes. On the other hand, unsupervised learning is used when labeled data is unavailable. The machine is trained on unlabeled data, and the model tries to predict the data on its own [12]. One area of machine learning that has grown significantly in recent years is Natural Language Processing (NLP). NLP aims to devise algorithms on how to understand human languages and generate replies that resemble those of humans. NLP has become a crucial component in applications such as chatbots, virtual assistants, and automatic translations [10]. Businesses can now reach a wider audience and engage with customers in a variety of languages. NLP has also made it possible to recognize sentiment in text data, which has greatly benefited the study of social media and customer service using chatbots and virtual assistants. NLP has also made it possible to create speech recognition technology leading to well-known applications such as Siri and Alexa. Text sentiment identification or text summarization have proved to be useful in social media, journalism, and education [13].

The emergence of Transformer model has been particularly useful as an efficient unifying model that has done well both in NLP and visual tasks. This has led to the development of Generative Pre-trained Transformers (GPTs), which were trained using massive language datasets like the Wikipedia Corpus and Common Crawl. The openAI's GPT APIs provide

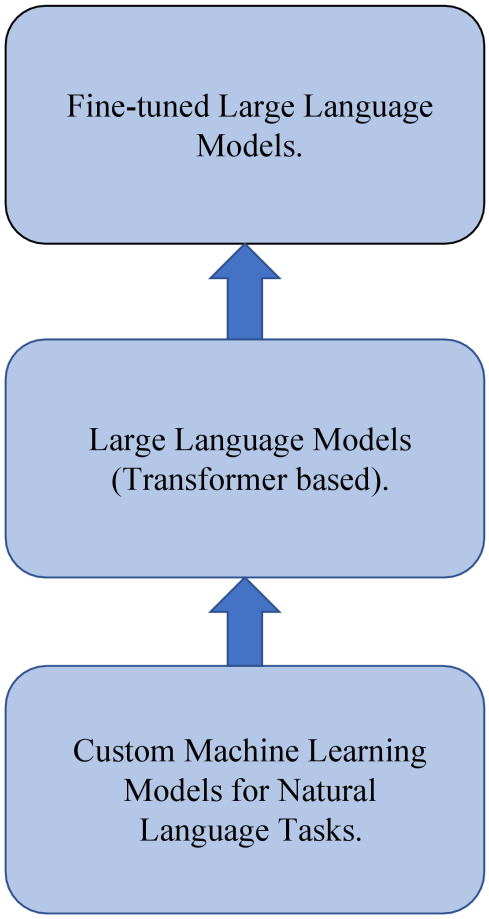

Fig. 1. Evolution Of Machine Learning.

access to the large language models trained on large corpora of text from various sources. GPT-3, which stands for Generative Pre-trained Transformer 3, is a state-of-the-art language model developed by OpenAI. Released in June 2020, it is the third iteration in the GPT series of models [10]. GPT-3 is a massive model, with 175 billion parameters and trained on a diverse range of internet text, allowing it to learn a vast amount of information and develop a strong understanding of language structure, grammar, and context. During the training phase, GPT-3 learns to generate text by predicting the next word in a sentence given its preceding context. There are different models within the GPT-3 category with varying levels of complexity, trained to do specific tasks better, and are available for fine-tuning such as davinci, curie, ada and babbage with varying price points for usage. As a point of reference, the cost for fine-tuning Ada model at the time of writing this is $0.0004 per 1000 tokens and for testing it is $0.0016 per 1000 tokens [3].

The likely futuristic evolution from an assorted collection of custom machine learning models to custom transformer models to fine-tuning large multi-modal models for language, visual, and audio tasks is exemplified in Figure 1 in the context of natural language processing. The scope of this paper is to use the GPT large language models to do zero-shot testing, and fine-tuning on a small set of training data of clickbait and non-click bait titles and compare the classification results both among the LLMs and also with previous work in the area. In doing so, we report that the power of the LLMs easily trump custom models in the prior work handily in various metrics including cost, training time, and accuracy. In the next section, we survey previous work in the literature on classification of titles as clickbait and non clickbait. In section III, we discuss the experimental setup, methodology, followed by results in section IV. In section V, we present some concluding observations and directions for further research.

II. PREVIOUS WORK

Previous research on identifying clickbait titles have used a variety of approaches and models [6], highlighting the challenging nature of the task to achieve high accuracy. [4]. Lahdji et al. [7] used a gradient-boosting classifier to train on 32,000 titles resulting in an accuracy of 95%. Chakraborty et al. [1], conducted a study to detect and prevent clickbait in online news media. The authors collected a dataset of 15,000 headlines from both clickbait and non-clickbait categories, and used the Stanford CoreNLP tool to carry out a detailed linguistic analysis. The authors used three different models (Support Vector Machines, Decision Trees, and Random Forests) to predict clickbait articles, and found that SVM performed the best with 93% accuracy, 0.95 precision, and 0.9 recall. They also developed a Chrome extension called ”Stop Clickbait” which alerts users about clickbait and gives them the option to block certain clickbait [1]. The extension was found to be effective, correctly blocking an average of 89% of tested links according to user feedback.

Shaikh and Annappanavar’s et al. [2], investigates clickbait detection by analyzing the textual characteristics of a database comprising over 50,000 articles, both clickbait, and non-clickbait. Text data is word-embedded into vectors and used as input to a CNN model using word2vec. The authors achieved good results with an F1 score of 0.86 and a precision of 0.82 [2]. The results are obtained in the form of a confusion matrix, which gives an accuracy of 82% [2].

The paper by Dimpas et al. [5], presents a different approach to clickbait detection in both Filipino and English languages using a BiLSTM (Bidirectional Long Short-Term Memory) neural network architecture. The dataset for the study comprises 5,000 clickbait and 5,000 non-clickbait headlines for both languages, collected from various web sources and websites based in the Philippines. The neural network model is built using the Keras API and Theano backend. The results of the study show that the proposed model achieved an accuracy of 93.50% for English and 92.87% for Filipino in detecting clickbait headlines. The authors highlight the significance of the BiLSTM architecture in the proposed model, as it takes into account the context of the words both before and after the current word, resulting in improved performance in clickbait detection. The study presents a promising approach

for clickbait detection in multiple languages and provides a useful contribution to the field of natural language processing [5].

Daoud et al. [8], proposed a supervised machine learning method to identify clickbait, using several variables based on text, structure, sentiment, and readability. The method employed machine learning techniques such as SVM, random forest, and naive Bayes to classify articles as clickbait or non-clickbait. The study achieved an F1 score of 79% and a ROC curve area of 0.7, indicating a high level of accuracy. The dataset consisted of 22,033 posts, with 2,495 posts used for training and the rest for validation. The authors utilized SVM, which takes each instance in the data set as a vector and plots it within a high dimensional space, dividing each class by building a hyperplane. By employing both the logistic regression and linear SVM techniques, they achieved an accuracy of 79%. Agrawal et al. [9], conducted a study to develop a model for clickbait detection using deep learning techniques. The model was built using handcrafted features from three fields: the teaser message or title, the linked web page, and the meta information. They utilized a basic CNN with just one convolutional layer, and the Click-Word2Vec model achieved an accuracy of 90% and an ROC-AUC of 0.90.

In our research, we explore various ways of using LLMs to improve the accuracy of detecting clickbait titles. In particular, we carry out zero-shot testing on both GPT 3.5 and GPT 4 models, followed by fine-tuning Ada with varying number of epochs, and ending with a novel iterative process of fine-tuning Ada with 4 epochs per iteration and increasing the training set size each time. In all of these, we find that our fine-tuning approach beats both zero-shot and results reported in the prior works above, yielding accuracy of between 96.5 to 99.5. In the next section, we present details of our research description and methodology.

III. RRESEARCH DESCRIPTION AND METHODOLOGY

A. Dataset and Software Tools

In order to carry out our research, we used the Kaggle dataset [14] which contains labeled titles of about 32,000 split roughly 50% of clickbait and non-clickbait titles, labeled as 1 or 0 respectively. We used Google Colab's Python as the computing environment to access OpenAI API and pandas library. The first step was to preprocess the dataset by converiting it to the *JSONL* format, as required by OpenAI API, using the *to_json* function of pandas. The *orient* parameter is set to *records* to ensure that the resulting JSON file contains one record per line, and the *lines* parameter is set to *True* to ensure that each record is written on a separate line. For clarity, JSONL is a variation of JSON that is designed to store multiple JSON objects in a more readable and scalable way. In JSONL, each line of the file contains a valid JSON object, and the objects are separated by newline characters. This format makes it easier to process large datasets line by line, allowing for more efficient streaming and processing of the data, especially when working with big data or log files.

B. Zero-Shot vs Fine-Tuned Models

OpenAI's data format for fine-tuning involves dividing the data into prompts and completions, where the prompt is the title and completion is true if the title is clickbait and false otherwise. The data is then converted to JSONL format, with column headings changed to "prompt" and "completion". In our case, we also converted the completion values from 0 to "False" and from 1 to "True". The resulting file must be in the *JSONL* format, which is the only format accepted by the API for fine-tuning. To prepare the data for input, we used OpenAI's library call *!openai tools fine_tunes.prepare_data -f file_name_jsonformat.jsonl -q* [3]. This added a '->' at the end of each prompt and added a space before the completion value. The data was automatically split into training and validation datasets for further analysis in our research.

```
import openai
def zero_shot_model(m):
 openai.api_key = "API_KEY_VALUE"
 model_engine = 'Models'
 response =openai.ChatCompletion.create(model=model_engine,\
                                        messages=[{"role":"system",\
                                                   "content":"Define_model_role"\
                                                   {"role":"user", \
                                                   "content":m}],\
                                                    n=1,\
                                                   temperature=0)
 message =(response['choices'][0]['message']['content'])
 return message
```

Fig. 2. Function for testing zero shot models.

We first use the base models of GPT-3.5 and GPT-4 natively to do the clickbait prediction as described below. A partial code of zero-shot is provided in Figure 2, where a function named *zero_shot_test* performs testing a sample random dataset using GPT-3.5 and GPT-4 models for clickbait detection. We provide a prompt through the *content*parameter, which is used by the model to generate completions. We define the model *engine* as either "gpt-3.5" or "gpt-4" depending on the model to be tested. We provide a prompt to the model that defines the task of the model. The prompt is *You are a helpful assistant. For each of the following user titles, say True if the title is a click-bait or say False otherwise. Each title is separated by ****. Your response should only be either True or False printed on a separate line*. The model is asked to determine whether a given user title is a clickbait or not by responding with either *True* or *False*. We include the *content* parameter in the prompt. Next, we call the *ChatCompletion.create()* method with the defined parameters to generate a completion from the model. The response from the model is saved to the variable *message*. Finally, the function returns the value generated by the model. The output is a string representing the model's prediction for each user title in the input, separated by a new line. We conducted experiments on a random dataset of 200 samples on models *gpt-3.5 turbo* and *gpt-4,* as Model A and Model B, respectively. The resulting accuracy was 73% and 88.5% respectively for A and B, as shown in Figure 5. One can see how good the base GPT-4 model is in detecting click-bait titles

without any fine-tuning at all and with a performance better than some of the custom models in prior work.

We further wanted to explore the influence of one of the hyperparameters namely, number of epochs. An epoch in the OpenAI API is one full pass through the training dataset. This means that each example in the dataset is presented to the model once. The number of epochs is a hyperparameter that can be tuned to improve the accuracy of the model. For example, if you are training a model on a dataset of 100,000 examples, and you set the number of epochs to 10, then the model will see each example 10 times. This will allow the model to learn more accurately, but it will also take longer to train. The optimal number of epochs depends on the size of the dataset, the complexity of the model, and the available resources. In this part of the research, the goal was to fine-tune the Ada model using 1000 clickbait and non-clickbait titles, which were fine-tuned for 4 and 20 epochs, respectively. OpenAI models have a default epoch value of 4. We generated two fine-tuned models of Ada one with 10 epochs and the other with 4 epochs using the 1000 sample training dataset called Model D and Model C respectively. The partial code in Figure 3 shows how to fine-tune a model. The fine_tunes.create system call is used to create the fine-tuned model, specifying the training and validation files to be used for fine-tuning with the -t and -v options, respectively. Correctly formating the training samples into OpenAI API compatible JSON is a crucial step. Here are two sample titles formatted in JSON.

- ”prompt”:”15 Stunning Gift Wrapping Ideas For The Minimalist In You”,”completion”:true
- ”prompt”:”Allotting of Iraqi Oil Rights May Stoke Hostility”,”completion”:false

```
import openai
import os
os.environ["OPENAI_API_KEY"] = "OPEN_API_KEY_VALUE"
!openai api fine_tunes.create \
      -t "file_name_prepared_train.jsonl" \
      -v "fine_name_prepared_valid.jsonl" \
      --compute_classification_metrics \
      --classification_positive_class " True" \
      -m ada  \
      --n_epochs 4
```

Fig. 3. Partial code for fine-tuning the Ada model.

To optimize the model's performance, we included hyperparameters such as *compute_classification_metrics* and *classification_positive_class='true'*. The former calculates accuracy, ROC, recall, and F1 scores for each model, while the latter sets the positive class to *true*. The *-m* option is used to specify which model to fine-tune. The *fine_tunes.follow* system command is used to track the fine-tuning process, while the *fine_tunes.cancel* system command can be used to cancel it if needed [3].

The accuracy of models generated were evaluated on an unseen dataset consisting of 200 samples. In Figure 4, we show the Completion.create() method to generate prediction as text completion. To create a completion, we passed parameters to the Completion. create() method which are the *engine*, specifying the fine-tuned model, *prompt* specifying the input prompt, *max_tokens* parameter specifying the maximum number of tokens that the model can generate in the completion, *logprobs* parameter specifying the number of log probabilities that the model returns for each token in the completion, and *temperature* parameter setting the randomness of the model's output. Using the results generated, a performance comparison of zero-shot models with the fine-tuned models is shown in Figure 5. Figure 7 shows the *confusion matrix* for Model D, illustrating the low number of FPs and FNs.

```
completions = openai.Completion.create(
    engine=fine-tuned_model,
    prompt=prompt,
    max_tokens=1,
    logprobs=2,
    temperature=0
)
```

Fig. 4. Partial code for testing a fine-tuned model.

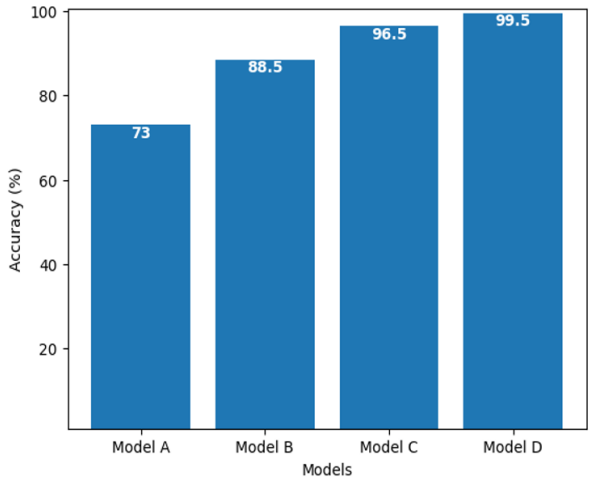

Fig. 5. Classification accuracy of Zero-Shot (A, B) and Fine-Tuned (C, D) Models.

C. Iterative Fine-Tuned Models

In the second approach, we iteratively fine-tuned the Ada LLM model starting with a dataset of 200 clickbait and non-clickbait title samples (roughly 50% each) for four epochs to get a new fine-tuned model, testing the model's accuracy on 200 unseen data samples, and increasing the data samples by 200 after each iteration. Then re-fine-tune for a new model and so on. We repeated this process five times, generating a new model after each fine-tuning iteration that we called Model 1.2, Model 1.4, Model 1.6, Model 1.8 and Model 2 respectively. The dotted decimal representation represents a way to indicate how many training samples were used. For instance, Model 1.2 refers to training with 200 samples, Model

1.4 refers to training with an additional 200 samples for a total of 400 samples, etc. Once again, we tested each model with a dataset of 200 samples. The prediction accuracy for the five iterated, fine-tuned models is shown in Figure 6.

IV. RESULTS AND COMPARISON

In Table 1 below, we have shown the models employed by previous authors drawing a comparison of the performance of such models with ours. Reviewing the table data, it is clear that fine-tuned LLMs perform better than custom models from previous search, and some by big margins. Our models beat existing ones by a margin of anywhere from 4.5% to 20%. It is also important to note that the improved performance comes by way of increased developer time to find the right tuning parameter contrasted with spending less development and testing time, a smaller set of training data, and a lower compute cost. Our OpenAI API usage cost a total of around 8 US dollars for all the models combined including training and testing.

Furthermore, within our research, the models show different capabilities. First, the zero-shot models of both GPT-25 and GPT-4 are already superior to some of the hand-crafted models [2], [8]. This confirms the power of the LLMs that we already have come to appreciate in the short period of time they have been around. However, Models C, D and the iterative, fine-tuned models demonstrate that there are real performance gains to be realized by massaging the LLMs with training data for the particular application. The fine-tuned models show a gain of between 5% to 10% over the zero-shot LLMs. We expect this to be the case for many of the previously studied NLP tasks, not just clickbait detection. One could conceivably revisit all prior NLP research and hope to rewrite the results. OpenAI has suggested that the default epoch of 4 is sort of where they expect the best fine-tuning to happen. This seems to be borne out in our experiments too, at least in this one application. For instance, Model D was fine-tuned with the default 4 epochs showing a slightly higher performance than Model C with 10 epochs. This just means that more epochs do not necessarily allow the fine-tuned model to generalize much more perhaps due to over-fitting.

TABLE I
COMPARISON OF CLICKBAIT DETECTION RESULTS

Authors	**ML Model**	**Clickbait Accuracy**
Daoud et al [8]	SVM, RF, NB	79%
Shaik et al [2]	CNN	82%
Agrawal et al. [9]	CNN	90%
Chakraborty et al. [1]	SVM	93%
Dimpas et al [5]	BiLSTM	93.5%
Lahdji et al. [7]	GB	95%
Our research	Zero-shot GPT-3.5-Turbo.	73%
-same-	Zero-shot GPT-4.	88.5%
-same-	Fine-tuned 3.5, 20 epochs	96.5%
-same-	Fine-tuned 3.5, 4 epochs	99.5%
-same-	Fine-tuned iterative	98.5%

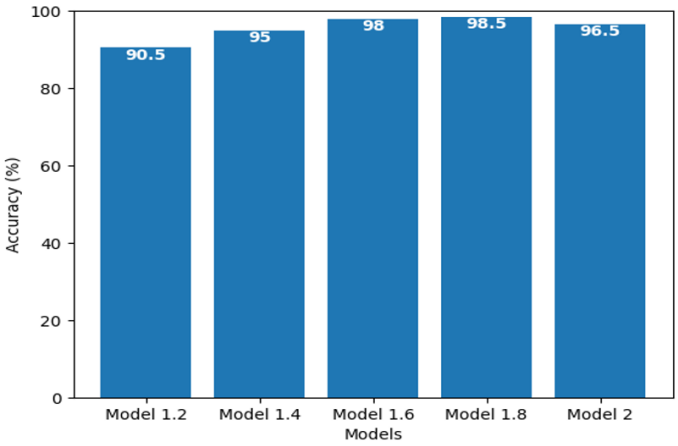

Fig. 6. Accuracy of Fine-tuned Models using the Iterative Approach

A. Data And Code Repository

The GitHub repository [15] contains both the data and code used for this research. The repository consists of the dataset files used for each model, test datasets, and code used for data pre-processing, fine-tuning, and model testing.

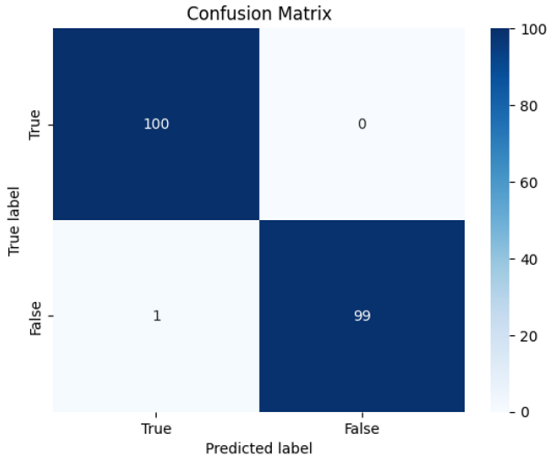

Fig. 7. Confusion matrix for Model D.

V. DISCUSSION AND CONCLUSION

Our research suggests that fine-tuning Open AI's large language model using substantially smaller training set and lower model development time, is an effective approach to detecting clickbait titles. Furthermore, we demonstrate that despite the widely-accepted notion that the LLMs (OpenAI) are already quite good in natural language understanding and interpretation, they still can be fine-tuned to improve their performance by a significant margin in custom tasks. We further observe that, the improvement in performance likely comes at a lower compute cost than previously studied approaches. Currently, there is still some effort that needs to go into tuning the hyperparameters or a following a different

approach like iterative refinement to improve performance. It is perhaps not inconceivable that the fine-tuning of an LLM itself may be learned using ML models in the future. We expect fine-tuned LLMs to show across-the-board gains on performance over custom NLP tasks that have been previously studied in the context of medicine, law, etc. Now, going beyond LLMs to large multi-modal models one can well imagine the tidal shift in future machine learning research.

REFERENCES

[1] Chakraborty, A., Paranjape, B., Kakarla, S., & Ganguly, N. (2016). "Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media". arXiv:1610.09786..

[2] M. A. Shaikh and S. Annappanavar, "A Comparative Approach For Clickbait Detection Using Deep Learning," 2020 IEEE Bombay Section Signature Conference (IBSSC), Mumbai, India, 2020, pp. 21-24

[3] OpenAI. Fine-Tuning - OpenAI API Documentation: https://platform.openai.com/docs/guides/fine-tuning

[4] P. Rajapaksha, R. Farahbakhsh and N. Crespi, "BERT, XLNet or RoBERTa: The Best Transfer Learning Model to Detect Clickbaits," in IEEE Access, vol. 9, pp. 154704-154716, 2021,

[5] P. Dimpas, R. Po and M. Sabellano, "Filipino and English clickbait detection using a long short-term memory recurrent neural network," International Conference on Asian Language Processing (IALP), pp. 276-280, 2017.

[6] S. Chawda, A. Patil, A. Singh and A. Save, "A Novel Approach for Clickbait Detection," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 1318-1321, doi: 10.1109/ICOEI.2019.8862781.

[7] Rayhan Lahdji. (2018). "Reusable Clickbait Detector: 95% CV Acc". https://www.kaggle.com/code/rayhanlahdji/reusable-clickbait-detector-95-cv-acc.

[8] Daoud, D., & ElSeoud, S. A. (2019). "An Effective Approach for Clickbait Detection Based on Supervised Machine Learning Technique". International Journal of Online and Biomedical Engineering (iJOE), 15(03), 21.

[9] A. Agrawal, "Clickbait detection using deep learning," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 268-272, doi: 10.1109/NGCT.2016.7877426.

[10] "GPT-3," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/GPT-3.

[11] "Transformer (machine learning model)," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model).

[12] IBM. (n.d.). Supervised vs. Unsupervised Learning. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

[13] [data-science-ua.com]. (n.d.). NLP: Achievements, Trends, and Challenges. Retrieved may,2023, from https://data-science-ua.com/blog/nlp-achievements-trends-and-challenges/

[14] Aman Anand Rai. (2019). Clickbait Dataset. https://www.kaggle.com/datasets/amananandrai/clickbait-dataset.

[15] pavansaivuppala. A-Fine-Tuned-Large-Language-Model-for-Improved-Click-Bait-Title-Classification [GitHub repository]. https://github.com/pavansaivuppala/A-Fine-Tuned-Large-Language-Model-for-Improved-Click-Bait-Title-Classification