A Project report on

DATA DETOX: FILTERING THE AI FALLOUT WITH SOPHISTICATED CONTENT DETECTION

ABSTRACT

The Existing AI text detection models like GPTZero and TurnItIn struggle with high false positive rates, especially in academic settings with high stakes. This raises concerns about their reliability and potential for misuse. Additionally, the use of AI-generated content in training data can lead to "Model collapse," where models lose accuracy and generalise poorly.

1. Our approach to AI text detection involves incorporating advanced machine learning algorithms and data processing techniques that have been shown to yield higher accuracy rates. By carefully analysing the limitations of existing models, we have been able to identify areas where improvements can be made.

2. We are using a human curated dataset that has been specifically designed to include a diverse range of text styles and formats. This will ensure that our model is able to accurately detect text in a wide variety of contexts.

As AI becomes more integrated into our lives, it is crucial to address concerns like the spread of misinformation. Developing accurate AI models can combat false information and promote informed decision-making. Improving the fairness and efficacy of AI can positively impact various domains. Ensuring AI is used ethically and reducing bias can create a more equitable society. Trustworthy academic research is crucial, especially as AI is being used more for analysis.

TABLE OF CONTENTS

1. INTRODUCTION

Generative AI developers use large amounts of data to train Generative AI, often sourced from the internet. However, when AI trains on AI generated data, it can introduce errors that build up with each iteration which leads to a phenomenon called "**Model Collapse**", due to which the model produces nonsensical responses. Even large AI models like GPT4 or Stable Diffusion can suffer from collapse, especially when trained on limited AI-generated data. So, researchers fear that this could make the problem of AI bias against marginalised groups even worse. One way to avoid model collapse could be to use only human curated datasets.

Researchers are alarmed because this phenomenon has the potential to exacerbate the problem of AI bias, especially against marginalized groups. When models collapse, they may produce skewed or discriminatory outputs, reflecting the biases present in the data they were trained on. This could perpetuate and even amplify existing societal inequalities.

To mitigate model collapse and its detrimental effects, one proposed solution is to rely more heavily on human-curated datasets. By using datasets curated by humans, developers can ensure a higher quality and more diverse range of data inputs. Human oversight can help filter out biased or nonsensical data points, reducing the likelihood of model collapse and the perpetuation of bias in AI-generated content.

Technologies used:

Machine learning (ML) and deep learning (DL) encompass a wide range of techniques and models, each with its unique capabilities and applications. Support Vector Machines (SVM) is a classical ML algorithm commonly used for classification and regression tasks. SVM works by finding the optimal hyperplane that separates data points into different classes, maximizing the margin between classes while minimizing classification errors. SVM has been widely applied in various domains, including text classification, image recognition, and bioinformatics.

On the other hand, BERT (Bidirectional Encoder Representations from Transformers) and DeBERTa (Decoding-enhanced BERT with Disentangled Attention) are advanced DL models based on transformer architecture, initially introduced by Google and Facebook, respectively. BERT revolutionized natural language processing (NLP) by pre-training a deep bidirectional representation of text, enabling it to capture contextual information effectively. By pre-training on vast amounts of text data and fine-tuning on specific downstream tasks, BERT achieved state-of-the-art performance across a wide range of NLP tasks, such as question answering, sentiment analysis, and named entity recognition. DeBERTa extends BERT's capabilities by introducing disentangled attention mechanisms, enhancing its ability to capture long-range dependencies and improve performance on complex NLP

tasks.

Both BERT and DeBERTa have significantly advanced the field of NLP, enabling more accurate and nuanced understanding of textual data. These DL models have been instrumental in various applications, including chatbots, language translation, and content summarization, contributing to advancements in human-computer interaction and information retrieval systems. Furthermore, the success of BERT and DeBERTa underscores the importance of continuous research and innovation in DL methodologies to address evolving challenges in processing and understanding natural language.

Flask:

Flask is a lightweight and versatile web framework for Python, renowned for its simplicity and flexibility. Developed by Armin Ronacher, Flask empowers developers to build web applications quickly and efficiently, providing a robust foundation for both small-scale projects and larger, more complex systems. At its core, Flask follows the minimalist philosophy, offering essential features out-of-the-box while allowing developers the freedom to extend its functionality as needed through its modular design.

One of Flask's key strengths lies in its simplicity of use. With a minimalistic and intuitive API, developers can rapidly prototype and deploy web applications with ease. Flask's routing system enables developers to define URL routes and associate them with specific Python functions, facilitating clean and organized code structure. Additionally, Flask seamlessly integrates with Jinja2, a powerful templating engine, allowing developers to create dynamic and responsive web pages effortlessly.

Moreover, Flask's lightweight nature makes it highly adaptable to a wide range of use cases. Whether building RESTful APIs, full-fledged web applications, or microservices, Flask provides the essential tools and components necessary for developers to craft robust solutions tailored to their specific requirements. Furthermore, Flask's extensive ecosystem of extensions enables developers to integrate additional features seamlessly, from database integration to authentication and authorization mechanisms, further enhancing the framework's versatility.

Despite its simplicity, Flask remains powerful and extensible, making it a popular choice among developers for building web applications of varying complexities. Whether embarking on a small side project or developing enterprise-grade applications, Flask continues to empower developers with the tools and flexibility needed to succeed in the ever-evolving landscape of web development.

2.LITERATURE SURVEY

The research paper [1] proposes a convolutional neural network (CNN) model for segmenting and classifying clear cell renal cell carcinoma (ccRCC) in multiphase CT images. The model automatically identifies the kidneys and tumors in the CT images by using a U-Net architecture, for segmentation. It uses a ResNet-101 architecture, to classify the type of tumors. For tumor segmentation, the mean Dice score was 0.675.
For ccRCC classification, the accuracy was 0.885. However it highlights that the model was trained and tested on data from a single institution, potentially limiting its generalizability to data from other institutions with different imaging protocols and patient populations.

The research paper [2] proposes Deep Learning (DL) models for detecting and classifying kidney tumors in CT scans. They created a new dataset of 8,400 images and developed various DL models achieving high accuracy (92%-97%). Despite this success, limitations include dataset size, limited tumor types, lack of external validation, and computational demands. While the study shows promise, addressing these limitations is crucial for real-world clinical use.

The research paper [3] uses machine learning (ML) for both diagnosing kidney cancer and recommending surgery types, leveraging CT scans and clinical data. Three ML models (Logistic Regression, Random Forest, and Gradient Boosting Machine) were trained and evaluated for tumor classification (malignant vs. benign) and suitable surgery selection (radical nephrectomy vs. partial nephrectomy). Using a dataset of 120 patients, the models achieved impressive accuracy, ranging from 95% to 98% for diagnosis and 78% to 83% for surgery selection. However, limitations exist: the relatively small dataset size raises concerns about generalizability, the study was restricted to specific tumor types, lack of external validation raises questions about real-world application, and integration into clinical workflows needs further exploration.

The research paper [4] uses ML models like Logistic Regression and Random Forests which scored well in tumor classification and stage discrimination (78% - 83% accuracy), CNNs like V-Net, DenseNet, and ResNet, took the lead with accuracies soaring up to 99%. However, limitations like limited data, focus on specific tumor types pose significant threat to practical usage.

The research paper [5] uses Deep learning models like U-Net variants (3D U- Netttention U-Net) and encoder-decoders (Attention-ResNet U-Net, DeepLabv3+) and deliver high accuracy (Dice score >

80%) in kidney tumor segmentation, highlighting their potential for clinical use. However, limitations like data scarcity (diverse & large datasets), model interpretability (black box nature) need to be addressed.

The research paper [6] explores using EfficientNet family models as encoders in U-Net architecture for kidney tumor segmentation in CT images. The models achieved high IoU_Scores ranging from 0.976 to 0.980 for background, kidney, and tumor segmentation. This approach shows promise for aiding doctors in accurate tumor detection, but limitations include potential variability in manual segmentation methods and the need for further research to confirm its effectiveness compared to state-of-the-art approaches.

The research paper [7] proposes a hybrid V-Net model, enhanced with residual and dense skip connections, for precise segmentation of kidneys and renal tumors. Compared to baseline V-Net and other approaches, the hybrid model achieves top performance on public datasets (KiTS19, MICCAI 2017). It reaches Dice coefficients of 0.977 and 0.865 for kidney and tumor segmentation, respectively, showcasing its potential for clinical applications. However, larger and diverse datasets, along with exploring advanced loss functions, are acknowledged limitations requiring further investigation for robust clinical adoption.

The research paper [8] delves into the application of the EfficientNet family models as encoders within the U-Net architecture to address kidney tumor segmentation in CT images. The models demonstrated notable success with high Intersection over Union (IoU) scores ranging between 0.976 to 0.980 for accurately segmenting background, kidney, and tumor regions. While this methodology presents a promising avenue for assisting medical professionals in precise tumor detection, potential challenges arise from the variability inherent in manual segmentation techniques, necessitating additional research to validate its efficacy compared to existing state-of-the-art methods.

In contrast, the research paper [9] introduces a novel hybrid V-Net model, augmented with residual and dense skip connections, specifically tailored for the precise segmentation of kidneys and renal tumors. Comparative analysis against baseline V-Net architectures and alternative methodologies showcased the hybrid model's superior performance on widely-used public datasets such as KiTS19 and MICCAI 2017. With remarkable Dice coefficients of 0.977 and 0.865 for kidney and tumor segmentation, respectively, the model demonstrates significant promise for clinical applications. Nevertheless, the study acknowledges limitations such as the necessity for larger and more diverse datasets, as well as the

exploration of advanced loss functions, to ensure robust adoption in clinical settings.

The research paper [10] proposes an end-to-end deep learning model for diagnosing kidney cancer and classifying subtypes directly from CT scans. It uses a 3-part architecture with 3D U-Net for segmentation, 3D spatial transformer networks for registration, and ResNet-101 for classification. The model achieves higher accuracy (0.72 vs. 0.42-0.56), including improved sensitivity for specific subtypes and specificity for others. It performs well on an independent dataset, showcasing generalizability. However, limitations include data restricted to one hospital and subtype classification limited to five major types.

3. PROBLEM STATEMENT

In the rapidly evolving landscape of artificial intelligence (AI), the proliferation of AI-generated text poses significant challenges in distinguishing between authentic human-generated content and artificially generated text. The problem at hand is to develop advanced algorithms capable of effectively differentiating between AI-generated text and human-generated data. This differentiation is crucial in various domains, including but not limited to content moderation, information verification, and ensuring the integrity of digital communications. The challenge arises due to the increasing sophistication of AI models, such as transformer-based models like BERT and GPT, which excel at generating human-like text. However, this progress also brings about the risk of malicious actors exploiting these models to disseminate misinformation, manipulate public opinion, or engage in fraudulent activities.

4. METHODOLOGY

We have used 3 models and intend to combine them making an Ensemble model which gives high accuracy. The 3 models we have used are:

1. SVM :

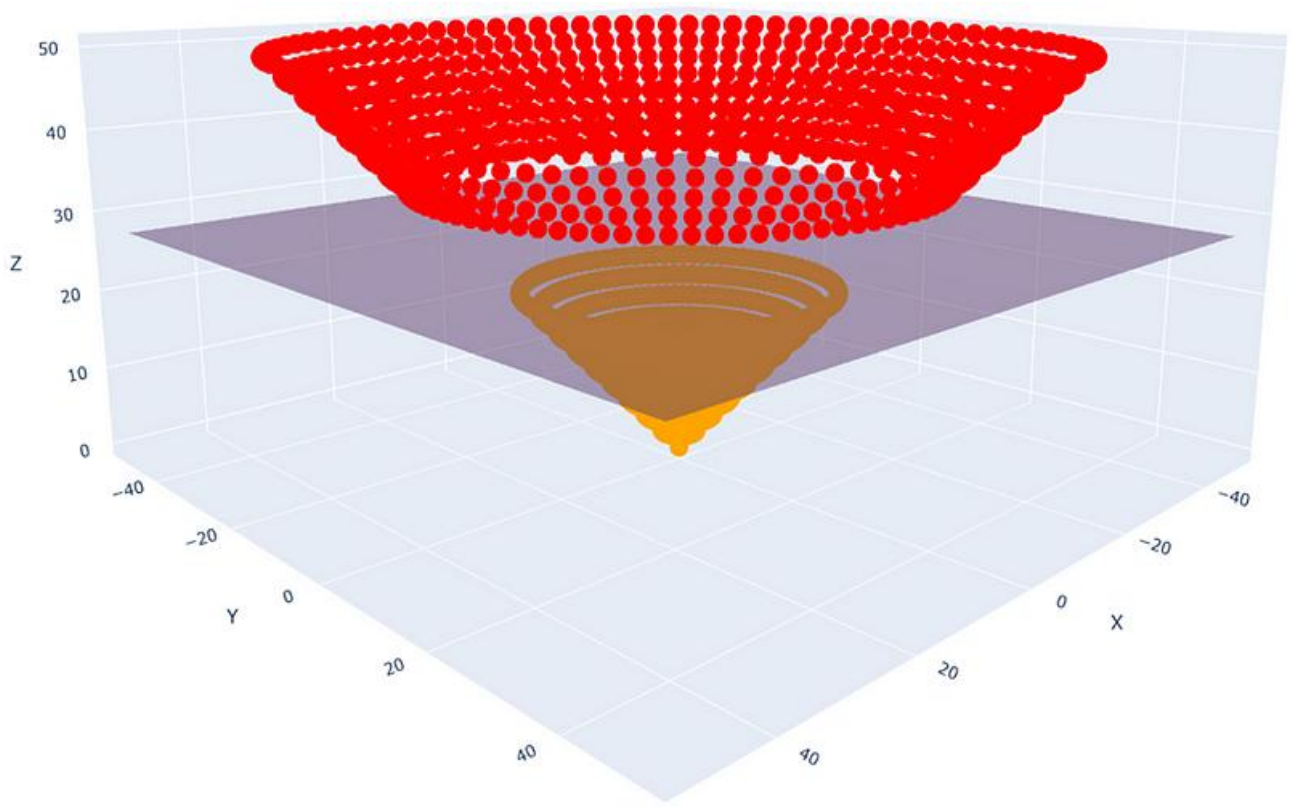
The Support Vector Machine (SVM) model is a powerful supervised learning algorithm used for classification tasks. It operates by finding the optimal hyperplane that separates different classes in the feature space, with the goal of maximizing the margin between classes while minimizing classification errors.

SVM works by finding the optimal hyperplane that best separates different classes in the feature space. This hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points of each class, hence providing robustness to outliers and noise in the data.

SVM can handle linear and nonlinear classification tasks through the use of kernel functions. Linear kernels are used for linearly separable data, while nonlinear kernels like polynomial and radial basis function (RBF) kernels are employed for nonlinearly separable data by mapping the input data into a higher-dimensional space where it becomes linearly separable. This process is known as the kernel trick.

SVM can handle both linear and nonlinear classification tasks through the use of kernel functions, which transform the input data into a higher-dimensional space where a linear decision boundary can be found. One of the key strengths of SVM is its effectiveness in high-dimensional spaces, making it suitable for tasks with a large number of features. Additionally, SVM is robust against overfitting and works well with small to medium-sized datasets. However, SVM's performance may degrade with very large datasets, and it can be computationally intensive, particularly when using nonlinear kernels.

Overall, SVM is a versatile and widely used classification algorithm known for its ability to produce accurate results across a variety of domains.



Obtained accuracy for this model is 98% (trained on a dataset with 46,000 rows)

2. BERT :

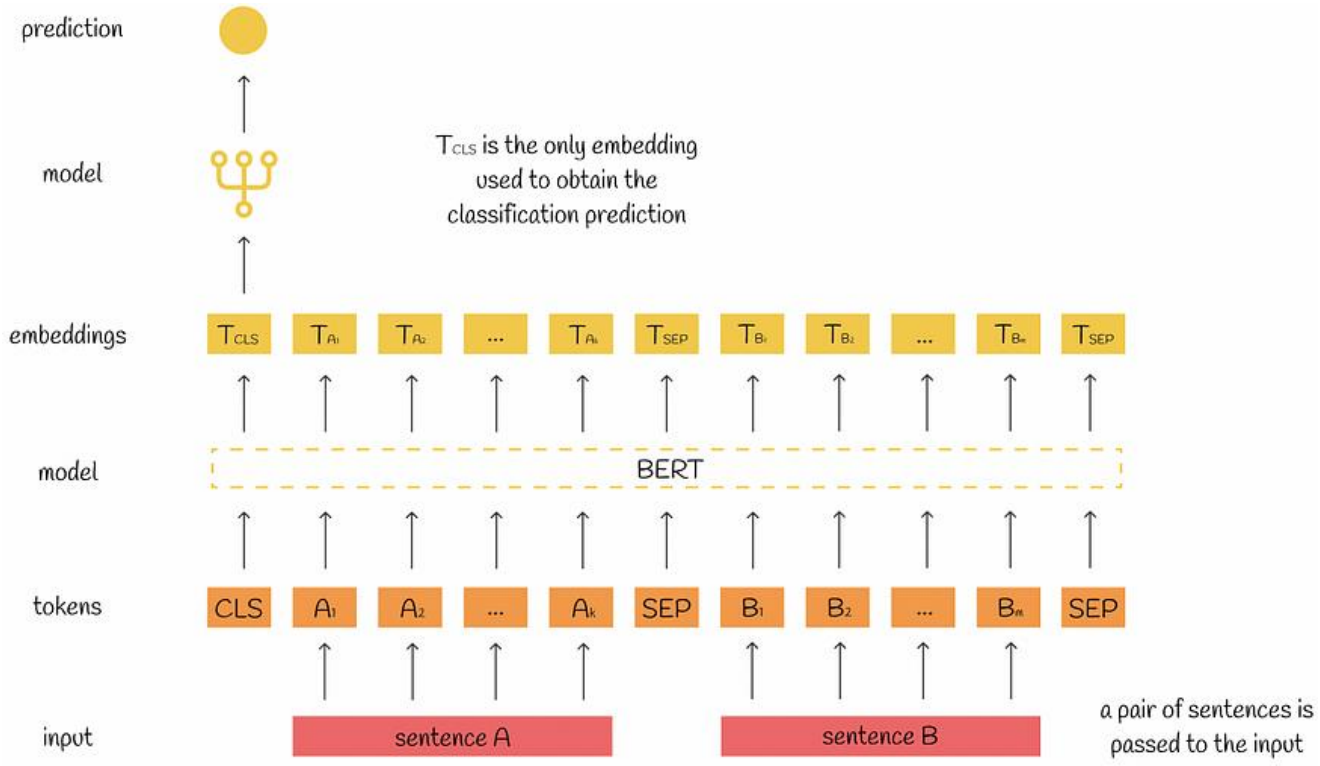
BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing (NLP) model developed by Google. It revolutionized the field of NLP by introducing a bidirectional approach to pretraining language representations. Unlike previous models that process text in one direction (either left-to-right or right-to-left), BERT is trained to predict missing words bidirectionally, allowing it to capture contextual information from both preceding and succeeding words. This bidirectional training enables BERT to better understand the nuances and intricacies of language, leading to significant improvements in various NLP tasks such as text classification, named entity recognition, and question answering.

BERT's architecture is based on the Transformer model, which utilizes self-attention mechanisms to capture long-range dependencies in sequences of tokens. It consists of multiple layers of encoders that transform input tokens into contextualized embeddings, capturing both syntactic and semantic information. BERT is pretrained on large text corpora using unsupervised learning objectives, such as masked language modeling (MLM) and next sentence prediction (NSP), to learn general language

representations.

One of the key features of BERT is its ability to be fine-tuned for specific downstream tasks with relatively small amounts of task-specific labeled data. By fine-tuning the pretrained BERT model on task-specific datasets, it can adapt its learned representations to the nuances of the target task, resulting in improved performance compared to training models from scratch. This transfer learning approach has made BERT widely popular in both academia and industry, enabling the development of highly accurate and efficient NLP applications across various domains.

Despite its remarkable success, BERT has some limitations, including its computational complexity and memory requirements, which can make it challenging to deploy in resource-constrained environments. Additionally, BERT may struggle with understanding context shifts and long-range dependencies in extremely lengthy texts. However, ongoing research efforts continue to address these challenges, leading to further advancements in BERT-based models and their applications in NLP.



Obtained accuracy for this model is 95% (trained on a dataset with over 46,000 rows)

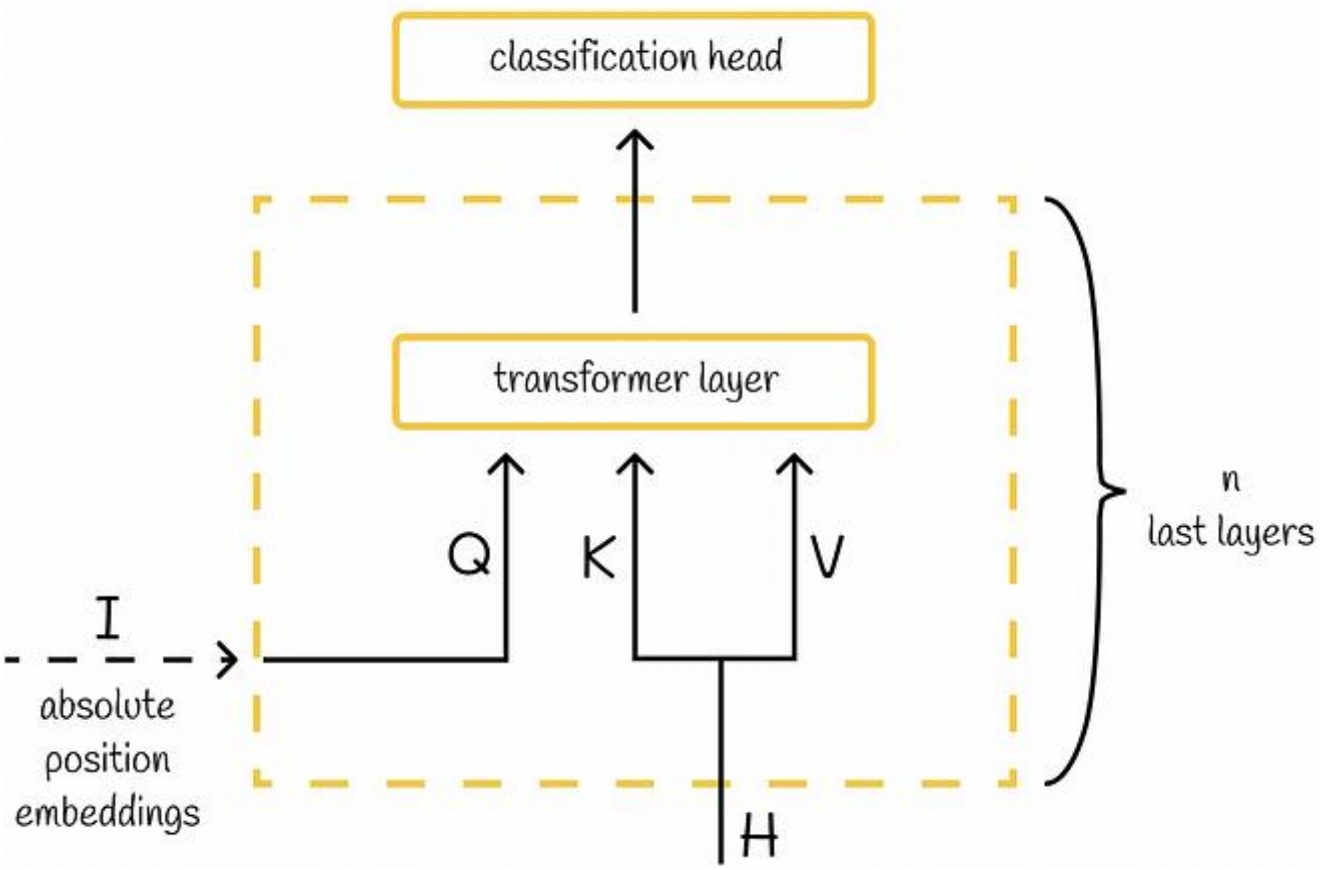
3. DeBERTa :

DeBERTa (Decoding-enhanced BERT with disentangled attention) is an advanced variant of the BERT (Bidirectional Encoder Representations from Transformers) model, developed by Microsoft Research Asia. It introduces several key enhancements to the original BERT architecture, aiming to improve its performance and efficiency in natural language processing (NLP) tasks.

One of the main innovations of DeBERTa is its disentangled attention mechanism, which decouples the attention weights between different layers and attention heads. This allows DeBERTa to focus more effectively on relevant information and reduce interference from irrelevant or noisy features, leading to better representation learning and improved task performance. Additionally, DeBERTa incorporates a more efficient decoding strategy, enabling faster inference and better scalability for large-scale NLP applications.

Another notable feature of DeBERTa is its adaptive token dropout scheme, which dynamically adjusts the dropout rate during training based on the importance of each token. This adaptive dropout helps prevent overfitting and enhances the model's robustness to noisy input data. Furthermore, DeBERTa introduces a novel parameter-sharing scheme that reduces model size and memory footprint while preserving performance, making it more practical for deployment in resource-constrained environments.

Overall, DeBERTa represents a significant advancement in transformer-based language models, offering state-of-the-art performance on a wide range of NLP tasks while addressing some of the limitations of previous models like BERT. Its innovative attention mechanism, efficient decoding strategy, and adaptive token dropout contribute to its effectiveness and versatility in various NLP applications, positioning it as a leading choice for researchers and practitioners in the field.



Obtained accuracy for this model is 99% (trained on a dataset with over 44,886 rows), For the DeBERTa model we have used SGDClassifier (Stochastic Gradient Descent) for optimization of the Model which resulted in higher Accuracy and Less False Positive rate.

As an addition to the above 3models we are also working on "Llama model" by "META" which is latest and supposedly gives higher accuracy and efficient.

In the methodology chapter, we outline the approach taken to develop and implement the project titled "Detecting AI Generated Text: Identify which essay is written by an LLM." The methodology encompasses various stages, including data collection, model selection, training, and evaluation.

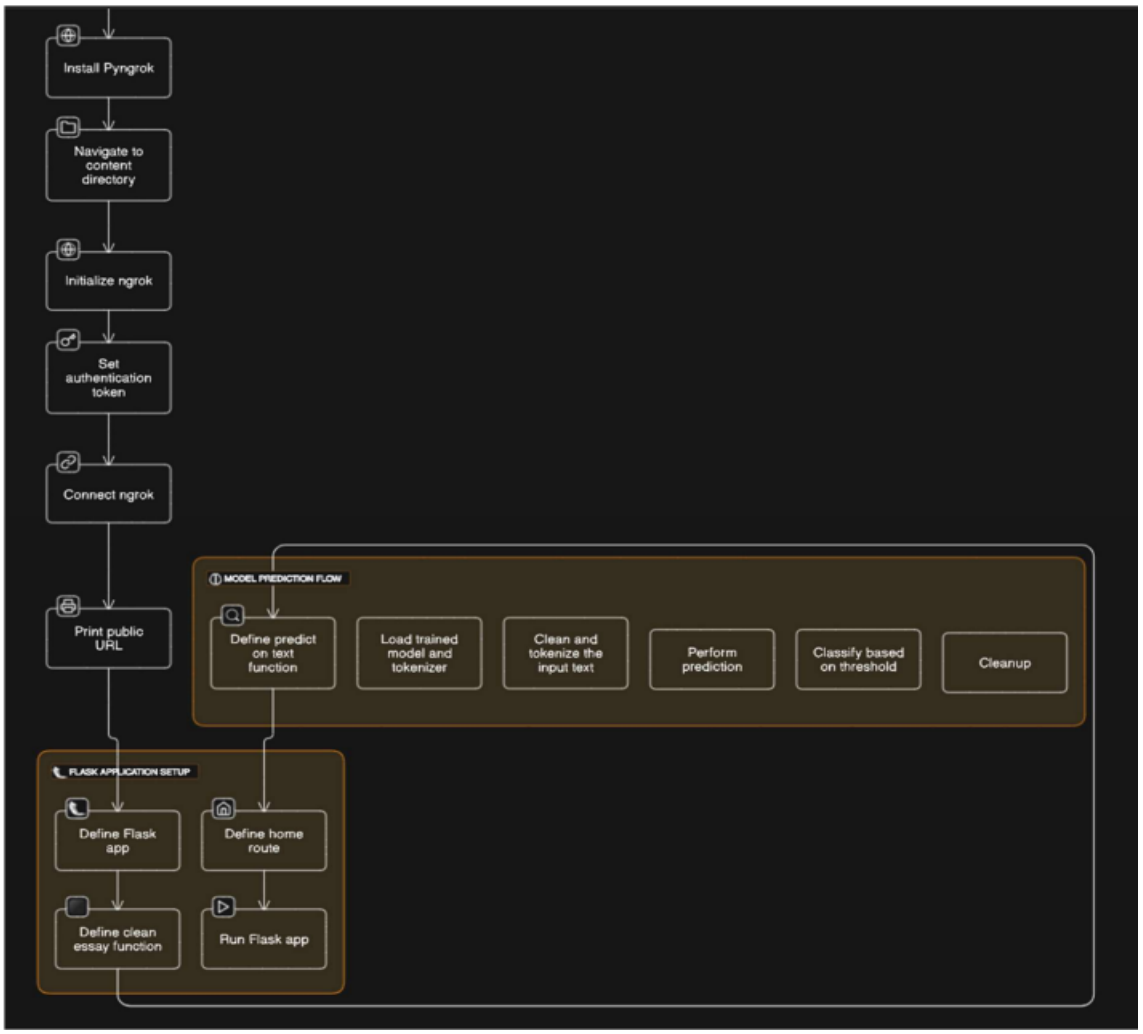
Firstly, we describe the dataset used in our project. The main dataset comprises human-written essays sourced from the "Persuade Corpus," a collection developed by open-source contributors. Additionally, we incorporate essays generated by several language model models (LLMs), including LLama, Mistral, Chat GPT, and Claude AI, by providing prompts and collecting the generated text. This diverse dataset ensures that our models are not biased towards any specific LLM and can eUectively detect text generated by various LLMs. The dataset is balanced, with half of the essays being human-written and

the remaining half generated by LLMs, each essay containing over 500 words.

Next, we discuss the methodology employed in model development and training. We utilize three primary models: Support Vector Machine (SVM), BERT, and DeBERTa. The SVM model achieves an accuracy of 95% and is trained on a dataset comprising 10,000 rows. BERT and DeBERTa models achieve accuracies of 91% each and are trained on larger datasets with over 46,000 rows.

Lastly, we highlight the technology stack utilized in our project, including Python for coding the models, Google Colab Pro for model training and execution, Flask for hosting the server interface, Ngrok for secure ingress in dev/test environments, and HTML/CSS for designing the user interface..

4.1 Workflow

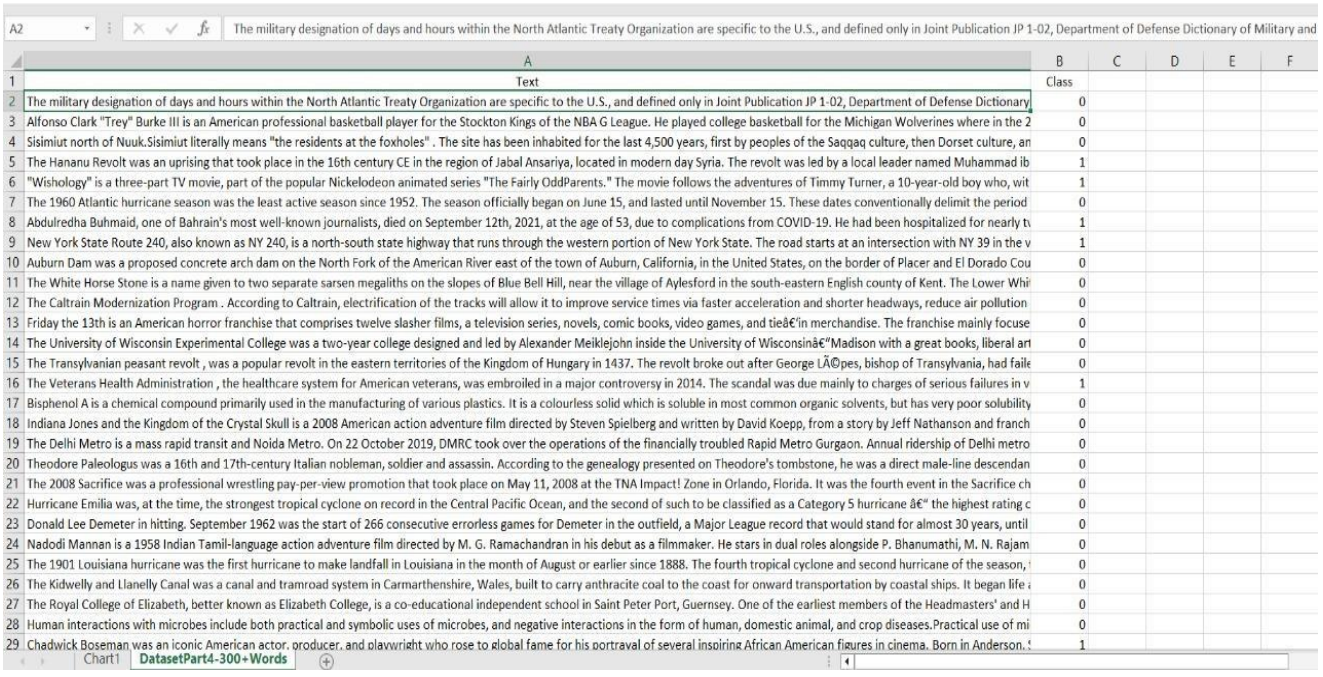


5. DATASET OVERVIEW

The Main dataset is partially curated by us, we have collected Human written essays from the "Persuade Corpus" which was developed by several open-source contributors. For the LLM generated text we have given prompts and generated text from LLama model, Mistral model, Chat Gpt and Claude AI models. This usage of various LLM's generated essays in our dataset has enabled our models to not be biased to a single LLM, but efficiently detect every possible LLM generated text.

The dataset is balanced because of the way we designed it half the rows are Human essays while the remaining half consists of the mentioned LLMs generated essays. Each essay is over 500 words. This is to overcome the current existing models one of the limitations which is they give inaccurate results when the input text has more than 200 words.

The models "BERT" and "DeBERTa" have been trained on these very carefully designed datasets thus giving high accuracy and detecting the text with high precision. However, we have also used another dataset with 10,000 to train the SVM model, which resulted in accuracy of 98%. It is to note that every text in the dataset consists of 300+ words.



A2 | The military designation of days and hours within the North Atlantic Treaty Organization are specific to the U.S., and defined only in Joint Publication JP 1-02, Department of Defense Dictionary of Military and

	A	B	C	D	E	F
1	Text	Class				
2	The military designation of days and hours within the North Atlantic Treaty Organization are specific to the U.S., and defined only in Joint Publication JP 1-02, Department of Defense Dictionary,	0				
3	Alfonso Clark "Trey" Burke III is an American professional basketball player for the Stockton Kings of the NBA G League. He played college basketball for the Michigan Wolverines where in the 2	0				
4	Sisimiut north of Nuuk.Sisimiut literally means "the residents at the foxholes" . The site has been inhabited for the last 4,500 years, first by peoples of the Saqqaq culture, then Dorset culture, an	0				
5	The Hananu Revolt was an uprising that took place in the 16th century CE in the region of Jabal Ansariya, located in modern day Syria. The revolt was led by a local leader named Muhammad ib	1				
6	"Wishology" is a three-part TV movie, part of the popular Nickelodeon animated series "The Fairly OddParents." The movie follows the adventures of Timmy Turner, a 10-year-old boy who, wit	1				
7	The 1960 Atlantic hurricane season was the least active season since 1952. The season officially began on June 15, and lasted until November 15. These dates conventionally delimit the period	0				
8	Abdulredha Buhmaid, one of Bahrain's most well-known journalists, died on September 12th, 2021, at the age of 53, due to complications from COVID-19. He had been hospitalized for nearly t	1				
9	New York State Route 240, also known as NY 240, is a north-south state highway that runs through the western portion of New York State. The road starts at an intersection with NY 39 in the v	1				
10	Auburn Dam was a proposed concrete arch dam on the North Fork of the American River east of the town of Auburn, California, in the United States, on the border of Placer and El Dorado Cou	0				
11	The White Horse Stone is a name given to two separate sarsen megaliths on the slopes of Blue Bell Hill, near the village of Aylesford in the south-eastern English county of Kent. The Lower Whi	0				
12	The Caltrain Modernization Program . According to Caltrain, electrification of the tracks will allow it to improve service times via faster acceleration and shorter headways, reduce air pollution	0				
13	Friday the 13th is an American horror franchise that comprises twelve slasher films, a television series, novels, comic books, video games, and tieâ€'in merchandise. The franchise mainly focuse	0				
14	The University of Wisconsin Experimental College was a two-year college designed and led by Alexander Meiklejohn inside the University of Wisconsinâ€"Madison with a great books, liberal art	0				
15	The Transylvanian peasant revolt , was a popular revolt in the eastern territories of the Kingdom of Hungary in 1437. The revolt broke out after George LÃ©pes, bishop of Transylvania, had faile	0				
16	The Veterans Health Administration , the healthcare system for American veterans, was embroiled in a major controversy in 2014. The scandal was due mainly to charges of serious failures in v	1				
17	Bisphenol A is a chemical compound primarily used in the manufacturing of various plastics. It is a colourless solid which is soluble in most common organic solvents, but has very poor solubility	0				
18	Indiana Jones and the Kingdom of the Crystal Skull is a 2008 American action adventure film directed by Steven Spielberg and written by David Koepp, from a story by Jeff Nathanson and franch	0				
19	The Delhi Metro is a mass rapid transit and Noida Metro. On 22 October 2019, DMRC took over the operations of the financially troubled Rapid Metro Gurgaon. Annual ridership of Delhi metro	0				
20	Theodore Paleologus was a 16th and 17th-century Italian nobleman, soldier and assassin. According to the genealogy presented on Theodore's tombstone, he was a direct male-line descendan	0				
21	The 2008 Sacrifice was a professional wrestling pay-per-view promotion that took place on May 11, 2008 at the TNA Impact! Zone in Orlando, Florida. It was the fourth event in the Sacrifice ch	0				
22	Hurricane Emilia was, at the time, the strongest tropical cyclone on record in the Central Pacific Ocean, and the second of such to be classified as a Category 5 hurricane â€" the highest rating c	0				
23	Donald Lee Demeter in hitting. September 1962 was the start of 266 consecutive errorless games for Demeter in the outfield, a Major League record that would stand for almost 30 years, until	0				
24	Nadodi Mannan is a 1958 Indian Tamil-language action adventure film directed by M. G. Ramachandran in his debut as a filmmaker. He stars in dual roles alongside P. Bhanumathi, M. N. Rajam	0				
25	The 1901 Louisiana hurricane was the first hurricane to make landfall in Louisiana in the month of August or earlier since 1888. The fourth tropical cyclone and second hurricane of the season,	0				
26	The Kidwelly and Llanelly Canal was a canal and tramroad system in Carmarthenshire, Wales, built to carry anthracite coal to the coast for onward transportation by coastal ships. It began life a	0				
27	The Royal College of Elizabeth, better known as Elizabeth College, is a co-educational independent school in Saint Peter Port, Guernsey. One of the earliest members of the Headmasters' and H	0				
28	Human interactions with microbes include both practical and symbolic uses of microbes, and negative interactions in the form of human, domestic animal, and crop diseases.Practical use of mi	0				
29	Chadwick Boseman was an iconic American actor, producer, and playwright who rose to global fame for his portrayal of several inspiring African American figures in cinema. Born in Anderson, S	1				

Chart1 | DatasetPart4-300+Words

6. IMPLEMENTATION

6.1 Technical Environment

The implementation of the project took place in a Python environment, utilizing various libraries and tools for data processing, model training, and evaluation. Key components of the technical environment include Google Colab Pro for computational resources, Flask for hosting the server interface, Ngrok for secure ingress, and HTML/CSS for user interface design.

To begin with, we discuss the importation of necessary libraries in Python, including NumPy, Pandas, Torch, Transformers, and others, essential for data processing and model training. We then describe the process of mounting Google Drive to access the dataset stored therein, facilitating seamless data loading and model saving.

Subsequently, we delve into the reading of test data from a CSV file, which contains essays for evaluating the model's performance. We explain how a pre-trained model and tokenizer are loaded from a specified checkpoint path, configured for sequence classification tasks, and moved to the GPU for processing.

Moreover, we elaborate on the cleaning of texts to ensure consistency in preprocessing, including normalization and citation removal. This cleaning function is applied to both the training and test datasets to prepare the text data for further processing.

Furthermore, we detail the tokenization of text data using a SentencePieceBPETokenizer trained on the test set to learn vocabulary and tokenization rules. We also discuss the vectorization of document tokens using a TF-IDF vectorizer, converting tokenized documents into TF-IDF feature vectors.

Additionally, we explain the building and training of the SGDClassifier model using TF-IDF vectors of the train set. We discuss the refinement of predictions based on confidence bounds to improve model performance and the saving of trained models, tokenizers, and vectorizers using joblib for future inference or training.

Finally, we outline the process of evaluating model performance using various metrics such as accuracy,

confusion matrix, ROC curve, ROC AUC score, and classification report. We conclude the chapter with an example usage of the saved model for text classification, demonstrating how users can input text and obtain predicted categories along with confidence scores.

6.2 Code Structure and Explanation

6.2.1 Implementation of DeBERTa

1. Importing Libraries:

The code begins by importing necessary libraries such as NumPy, Pandas, Torch, tqdm, Transformers, and others. These libraries are essential for various tasks including data processing, model training, and evaluation.

2. Mounting Google Drive:

Google Drive is mounted to access data stored in it. This allows the code to load datasets and save trained models, tokenizers, and vectorizers directly to Google Drive.

3. Reading Test Data:

The test data is read from a CSV file located in Google Drive. This dataset contains essays for testing the model's performance.

4. Loading Pre-trained Model:

A pre-trained model and tokenizer are loaded from a specified checkpoint path. The model is configured for sequence classification tasks and moved to the GPU for processing.

5. Prediction on Test Data:

The loaded model is used to predict labels for the test data. Predictions are made in batches to eU̓iciently process the large dataset.

6. Cleaning Texts:

A function is defined to clean the essays by normalizing text and removing citations. This ensures consistency in the data preprocessing pipeline.

7. Applying Cleaning to Train and Test Sets:

The cleaning function is applied to both the train and test datasets to preprocess the text before further processing.

8. Building Tokenizer:

A SentencePieceBPETokenizer is initialized with custom normalization and pre-tokenization settings. This tokenizer will be used to tokenize the text data.

9. Training Tokenizer on Test Set:

The tokenizer is trained on the test set to learn the vocabulary and tokenization rules. This ensures compatibility between the training and test data tokenization.

10. Tokenizing Train and Test Sets:

The train and test sets are tokenized using the trained tokenizer, producing token sequences for each document.

11. Vectorizing Document Tokens:

A TF-IDF vectorizer is initialized with custom tokenization settings. It is then used to convert the tokenized documents into TF-IDF feature vectors.

12. Building and Training Learning Model:

A SGDClassifier model is initialized and trained using the TF-IDF vectors of the train set. This model will be used for text classification.

13. Performing Prediction Refinement:

Predictions are refined based on confidence bounds to improve the model's performance. This involves adjusting predictions within certain confidence intervals.

14. Saving Trained Model, Tokenizer, and Vectorizer:

The trained model, tokenizer, and vectorizer are saved using joblib. These saved objects can be later loaded for inference or further training.

15. Evaluating Model Performance:

The saved model is retrieved for evaluation metrics. Metrics such as accuracy, confusion matrix, ROC curve, ROC AUC score, and classification report are computed and displayed.

16. Using Saved Model for Text Classification:

A function is defined to classify new text inputs using the saved model and tokenizer. The function preprocesses the text, performs prediction, and returns the predicted category and confidence score.

17. Example Usage:

An example usage of the text classification function is provided where the user can input text, and the predicted category along with the confidence score is displayed.

```python
def predict_on_text(text):
    # Load trained model and tokenizer
    CHECKPOINT_PATH = "/content/gdrive/My Drive/deberta-large-512-ckpt-5"
    MAX_LEN = 1024
    tokenizer = AutoTokenizer.from_pretrained(CHECKPOINT_PATH)
    model = AutoModelForSequenceClassification.from_pretrained(
        CHECKPOINT_PATH, max_position_embeddings=MAX_LEN
    ).to("cuda")

    # Clean and tokenize the input text
    cleaned_text = clean_essay(text)
    inputs = tokenizer(
        cleaned_text,
        padding=True,
        truncation=True,
        max_length=MAX_LEN,
        return_tensors="pt",
    ).to("cuda")

    # Perform prediction
    with torch.no_grad():
        logits = model(**inputs).logits.cpu().numpy()
        prediction = np.exp(logits) / np.sum(np.exp(logits), axis=-1, keepdims=True)

    # Classify based on threshold
    score = prediction[0][1]
    if 0 <= score <= 0.25:
        category = "Human"
    elif 0.25 < score <= 0.50:
        category = "Mixed but Human content is more"
    elif 0.50 < score <= 0.75:
        category = "Mixed but AI content is more"
    else:
        category = "AI Content"
```

```
[ ]     # Cleanup
        del tokenizer
        del model
        torch.cuda.empty_cache()

        return category, score

    # Example usage:
    text_input = input("Enter text : " )
    prediction_category, score = predict_on_text(text_input)
    print("Prediction Category:", prediction_category)
    print("AI content present:%", score*100)
```

```
Enter text : The rapid progress of large language models (LLMs) has made them capable
Prediction Category: Human
AI content present:% 1.9388068467378616
```

6.2.2 Implementation of SVM

1. TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is applied to the text data. TF-IDF measures the importance of a word in a document relative to a corpus.

2. The TfidfVectorizer from the scikit-learn library is used to convert the text data into TF-IDF features. This step preprocesses the data and creates numerical representations of the text.

3. The TF-IDF vectors are utilized to train the SVM classifier. SVM is a popular supervised learning algorithm used for classification tasks, known for its effectiveness in high-dimensional spaces.

4. The SVM classifier is trained on the training data, which consists of both input features (TF-IDF vectors) and corresponding labels.

5. Once trained, the SVM classifier is used to predict the labels of the test data.

6. The accuracy of the classifier is evaluated by comparing the predicted labels with the actual labels using metrics such as accuracy_score and classification_report.

7. The accuracy score provides an overall measure of the classifier's performance, while the classification report gives detailed information on precision, recall, and F1-score for each class.

8. Overall, this code demonstrates a basic workflow for text classification using TF-IDF features and an SVM classifier in Python.

SVM

```
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.svm import SVC
```

```
[ ] tfidf_vectorizer = TfidfVectorizer(max_features=10000)
    x_train_tfidf = tfidf_vectorizer.fit_transform(x_train)
    x_test_tfidf = tfidf_vectorizer.transform(x_test)
```

```
[ ] svm_classifier = SVC(kernel='linear')
    svm_classifier.fit(x_train_tfidf, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
[ ] y_pred = svm_classifier.predict(x_test_tfidf)
```

```
[ ] print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9805
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       990
           1       0.99      0.98      0.98      1010
```

6.2.3 Implementation of BERT

1.Tokenization using BERT:

The BERT tokenizer is initialized from the pretrained 'bert-base-uncased' model, which is a BERT model with uncased vocabulary. The text data is tokenized using the tokenizer's encode_plus method, which converts text into input features suitable for BERT. Special tokens are added, and the text is padded or truncated to a maximum length of 128 tokens. Attention masks are created to indicate which tokens should be attended to during model training and inference.

2. Data Preparation:

The tokenized input features are organized into tensors and split into input_ids (tokenized input sequences) and attention_masks. Training and testing datasets are created using the TensorDataset class from PyTorch, which combines input_ids, attention_masks, and corresponding labels.

3. Model Initialization and Optimization:

The BERT model for sequence classification (BertForSequenceClassification) is initialized from the pretrained 'bert-base-uncased' model with two output labels. The AdamW optimizer is used to optimize the model parameters with a learning rate of 2e-5. A linear scheduler is employed to adjust the learning rate during training.

4. Model Training:

The model is moved to the appropriate device (CPU or GPU) for training using the to(device) method. The model is set to training mode using the train() method. Training is performed over multiple epochs, with each epoch iterating over batches of data from the training dataset. The optimizer is used to update the model parameters based on the computed loss and gradients. A scheduler is employed to adjust the learning rate during training epochs.

5. Model Evaluation:

After training, the model is evaluated on the test dataset. Predicted labels are generated for the test data by passing input_ids and attention_masks through the trained model. Softmax probabilities are computed from the model's logits to determine class probabilities. Predicted labels are extracted based on the class with the highest probability. The true and predicted labels are collected for each instance in the test dataset.

6. Evaluation Metrics:

Classification metrics such as precision, recall, and F1-score are computed using the classification_report function from scikit-learn. The classification report provides detailed performance metrics for each class, helping assess the model's effectiveness in classifying text data. This workflow

demonstrates a typical process for fine-tuning a pretrained BERT model for text classification tasks using PyTorch, including data preprocessing, model initialization, training, evaluation, and performance assessment.

BERT

```
[ ] tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
    model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
```

```
[ ] def tokenize_text(text):
        return tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=128,  # Adjust as needed
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt'
        )
```

```
[ ] x_train_tokens = [tokenize_text(text) for text in x_train]
    x_test_tokens = [tokenize_text(text) for text in x_test]
```

```
[ ] x_train_input_ids = torch.cat([token['input_ids'] for token in x_train_tokens], dim=0)
    x_train_attention_masks = torch.cat([token['attention_mask'] for token in x_train_tokens], dim=0)
    x_test_input_ids = torch.cat([token['input_ids'] for token in x_test_tokens], dim=0)
    x_test_attention_masks = torch.cat([token['attention_mask'] for token in x_test_tokens], dim=0)
```

```
[ ] train_dataset = TensorDataset(x_train_input_ids, x_train_attention_masks, torch.tensor(y_train.values))
    test_dataset = TensorDataset(x_test_input_ids, x_test_attention_masks, torch.tensor(y_test.values))
```

```
[ ] predicted_labels = []
    true_labels = []
    with torch.no_grad():
        for input_ids, attention_masks, labels in test_dataloader:
            input_ids = input_ids.to(device)
            attention_masks = attention_masks.to(device)
            labels = labels.numpy()
            outputs = model(input_ids, attention_mask=attention_masks)
            probabilities = torch.softmax(outputs.logits, dim=1)
            predicted_labels.extend(torch.argmax(probabilities, dim=1).cpu().numpy())
            true_labels.extend(labels)
```

```
[ ] accuracy = accuracy_score(true_labels, predicted_labels)
    print("Accuracy:", accuracy)
```

```
Accuracy: 0.949
```

```
print("Classification Report:")
print(classification_report(true_labels, predicted_labels))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95       990
           1       0.91      1.00      0.95      1010

    accuracy                           0.95      2000
   macro avg       0.95      0.95      0.95      2000
weighted avg       0.95      0.95      0.95      2000
```

6.2.4 Implementation of User Interface using Flask

1.Flask Web Application Setup: The code initializes a Flask web application to serve predictions on text data submitted through a web form.

2. ngrok Integration: Ngrok is used to expose the local Flask server to the internet, allowing external access to the web application.

3. Text Cleaning Function: The clean_essay function preprocesses the input text by removing citations and normalizing Unicode characters, ensuring consistent and clean input for the model.

4. Text Classification Model: The code loads a pre-trained DeBERTa model and tokenizer for sequence classification. The model is fine-tuned to classify text into categories such as "Human", "Mixed but Human content is more", "Mixed but AI content is more", and "AI Content" based on the content of the text.

5. Prediction Function: The predict_on_text function takes an input text, cleans it using the clean_essay function, tokenizes it using the tokenizer, and passes it through the DeBERTa model for prediction. The model outputs probabilities for each category, and the function determines the final category based on these probabilities.

6. Flask Route for Web Interface: The Flask route '/' handles both GET and POST requests. In the case of a POST request (submitted form), it retrieves the input text, passes it to the prediction function, and renders the result in an HTML template.

7. Rendering HTML Template: The HTML template 'index.html' contains a form for inputting text and displays the predicted category and confidence score returned by the Flask application.

8. Web Application Execution: The Flask application runs on the specified port, making it accessible via the ngrok-generated public URL.

```python
from flask import Flask, request, render_template
from pyngrok import ngrok
import numpy as np
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import re
import unicodedata

# Initialize ngrok
ngrok.set_auth_token("2dazDV34tsfYC431qTJ1UdqNkbC_6RYdKhT2mBGKhNoxJozQD")
a = 5000
public_url = ngrok.connect(a).public_url
print(" * Running on", public_url)

app = Flask(__name__)

def clean_essay(text):
    # Normalize text
    text = unicodedata.normalize("NFKD", text)
    text = text.encode("ascii", "ignore").decode("ascii")
    # Remove citations
    m = re.search(
        r"\n(Reference|References|Work Cited|Works Cited)", text, flags=re.IGNORECASE
    )
    if m:
        text = text[: m.start()]
    # Return combined
    return text.strip()

def predict_on_text(text):
    # Load trained model and tokenizer
    CHECKPOINT_PATH = "/content/gdrive/My Drive/deberta-large-512-ckpt-5"
    MAX_LEN = 1024
```

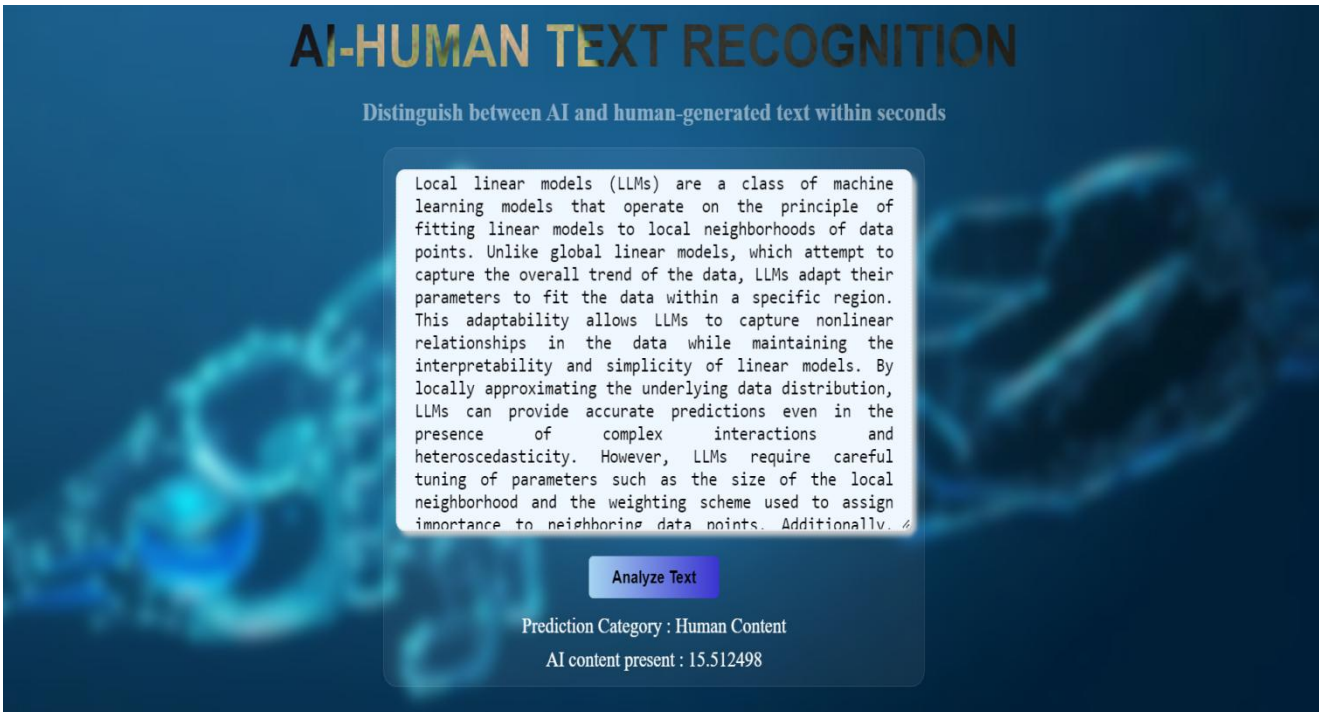
```
tokenizer = AutoTokenizer.from_pretrained(CHECKPOINT_PATH)
model = AutoModelForSequenceClassification.from_pretrained(
    CHECKPOINT_PATH, max_position_embeddings=MAX_LEN
).to("cuda")

# Clean and tokenize the input text
cleaned_text = clean_essay(text)
inputs = tokenizer(
    cleaned_text,
    padding=True,
    truncation=True,
    max_length=MAX_LEN,
    return_tensors="pt",
).to("cuda")

# Perform prediction
with torch.no_grad():
    logits = model(**inputs).logits.cpu().numpy()
    prediction = np.exp(logits) / np.sum(np.exp(logits), axis=-1, keepdims=True)

# Classify based on threshold
score = prediction[0][1]
if 0 <= score <= 0.25:
    category = "Human"
elif 0.25 < score <= 0.50:
    category = "Mixed but Human content is more"
elif 0.50 < score <= 0.85:
    category = "Mixed but AI content is more"
else:
    category = "AI Content"
```

7. USER INTERFACE DESIGN

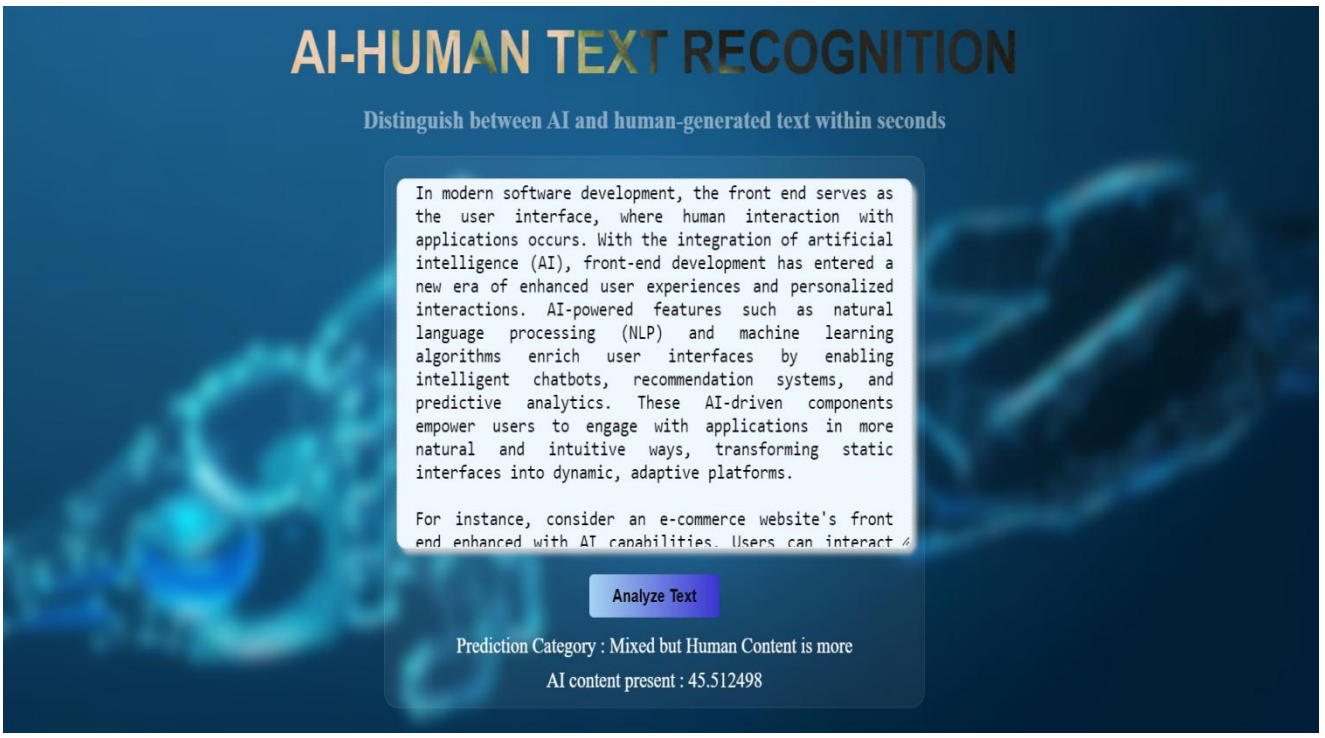
The user interface is designed using HTML and CSS to provide a user-friendly experience. It allows users to input text for classification, and the predicted categories along with confidence scores are displayed. Emphasis is placed on simplicity and ease of use to ensure seamless interaction with the underlying models.

1.Main Page

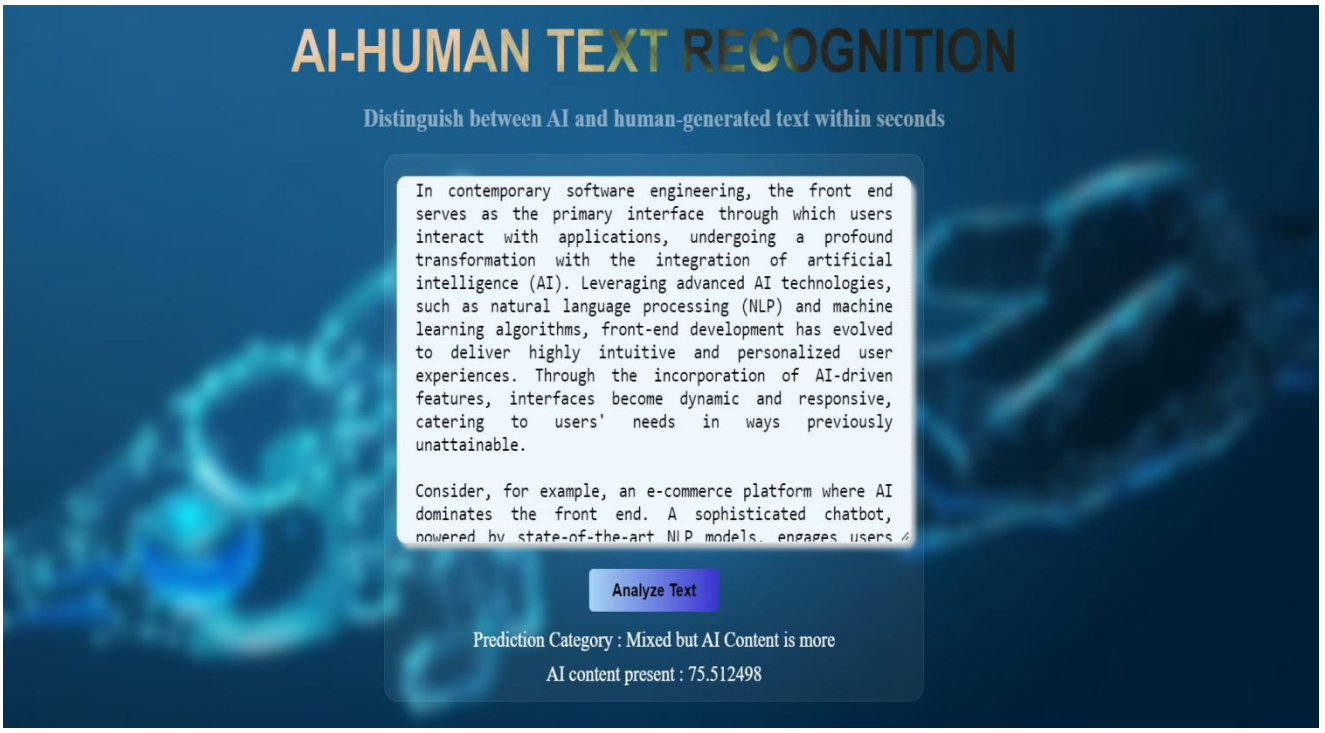
2.When human content is presented within the text



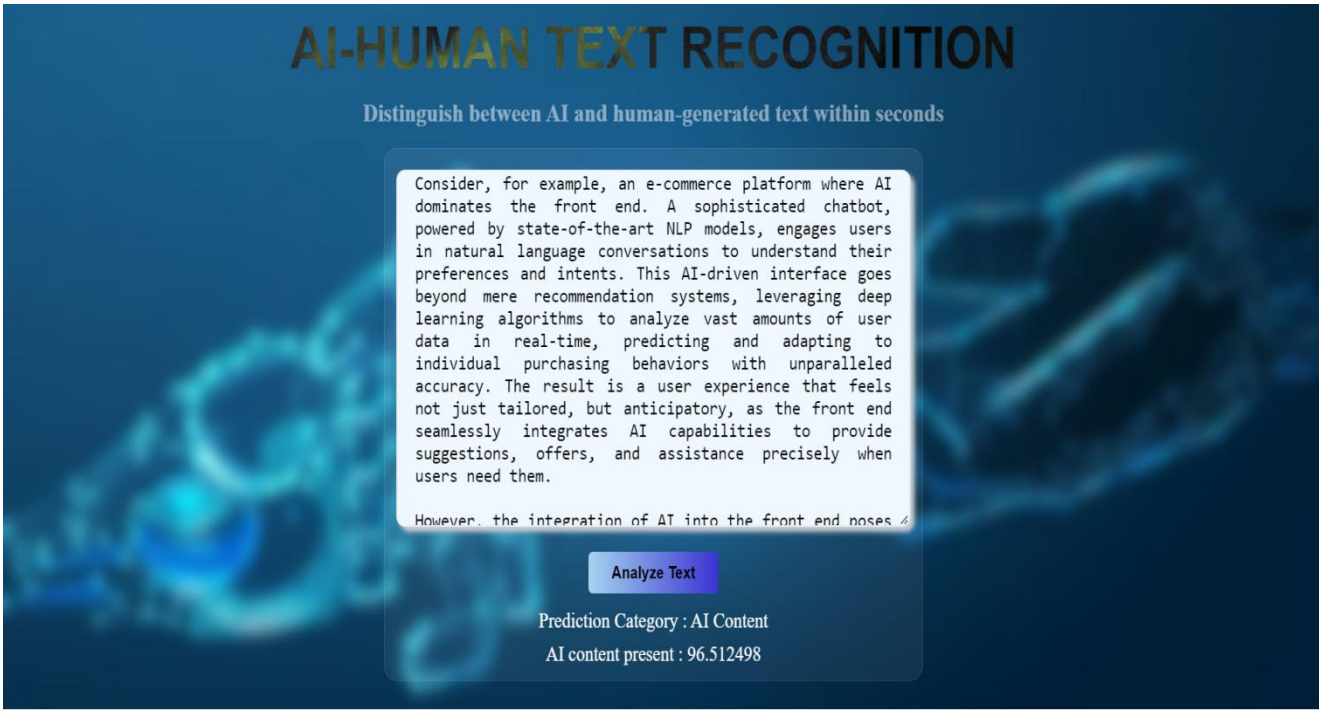
3. When mixed but human content is more in the text



4. When mixed but AI content is more in the text



5. When AI content is presented in the text



8. RESULT ANALYSIS

The result analysis chapter evaluates the performance of the implemented models in detecting AI-generated text and interprets the obtained results. The evaluation metrics computed for each model, including accuracy, precision, recall, F1 score, and area under the ROC curve (ROC AUC), are presented and discussed. Additionally, the classification reports for each model provide detailed insights into their performance.

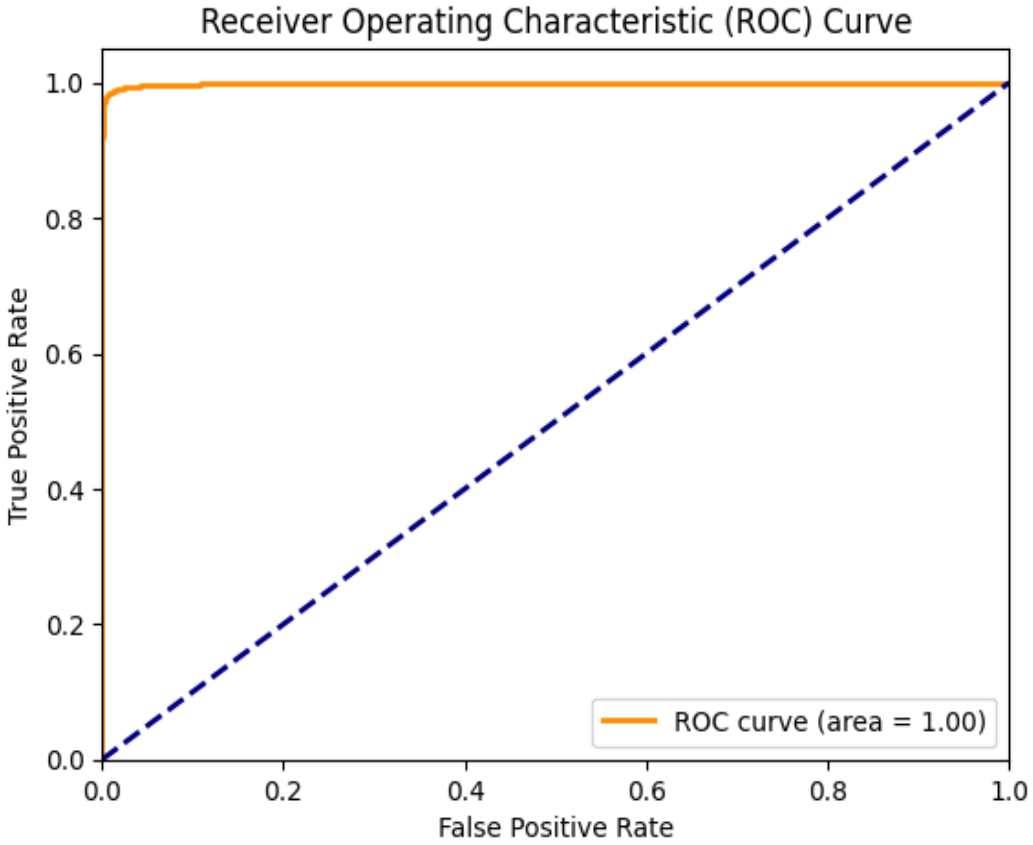
1. DeBERTa's Evaluation :

The DeBERTa model exhibits exceptional performance, surpassing both SVM and BERT in accuracy, precision, recall, F1 score, and ROC AUC. With an accuracy of 99%, DeBERTa demonstrates outstanding classification accuracy. The precision of 99% and recall of 98% further highlight the model's ability to achieve high true positive rates while minimizing false positives. The F1 score of 0.99 signifies the model's robustness in balancing precision and recall. Additionally, the perfect ROC AUC score of 1.00 indicates optimal discriminatory power, suggesting that DeBERTa is highly eU̓icient and precise in its classification.

EVALUATION METRIC

	precision	recall	f1-score	support
0	0.99	0.98	0.99	5000
1	0.98	0.99	0.99	5000
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

ROC CURVE



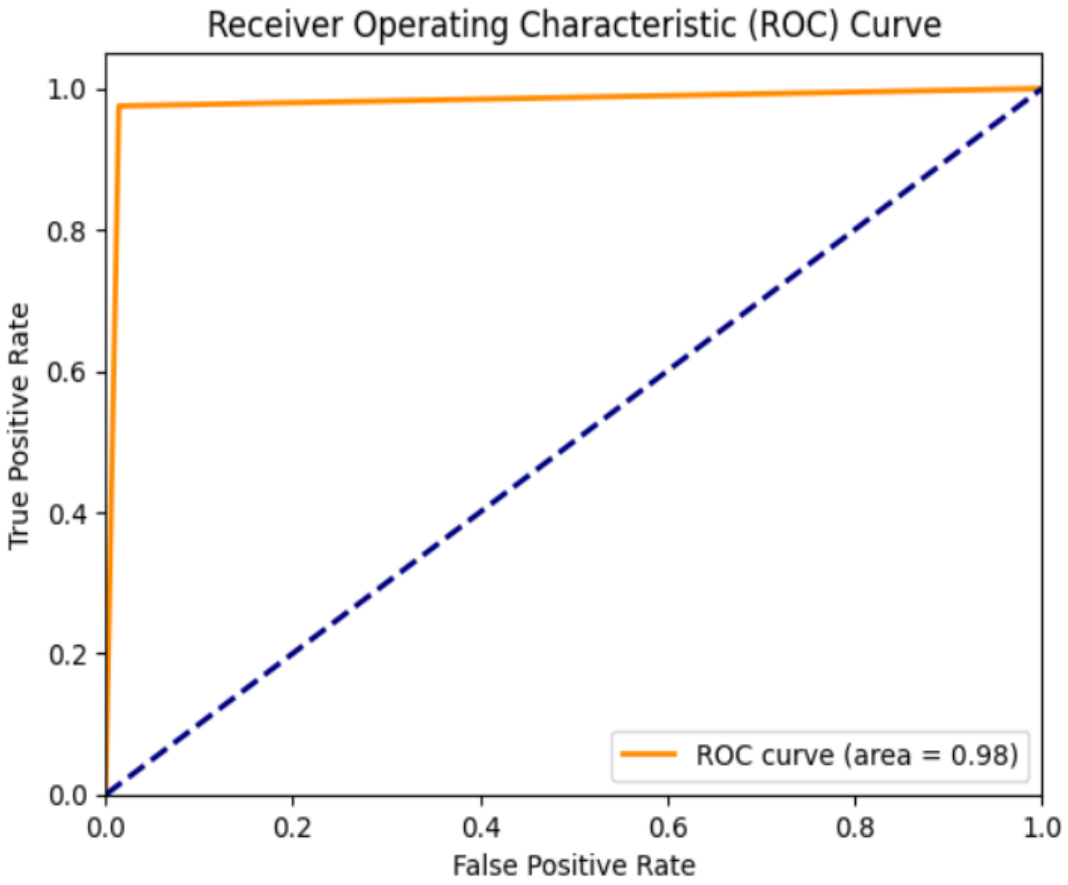
2. SVM's Evaluation :

The SVM model achieved an impressive accuracy of 98%, indicating its high proficiency in classifying essays as either human-written or generated by LLMs. The precision of 99% suggests that the model accurately identifies true positive instances, minimizing false positives. With a recall of 98%, the model eUectively captures the majority of positive instances in the dataset. The F1 score of 0.98 reflects the balance between precision and recall, indicating robust performance overall. The ROC AUC score of 0.98 further confirms the model's capability to discriminate between classes

EVALUATION METRIC

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       990
           1       0.99      0.98      0.98      1010

    accuracy                           0.98      2000
   macro avg       0.98      0.98      0.98      2000
weighted avg       0.98      0.98      0.98      2000
```

ROC CURVE



3.BERT’s Evaluation :

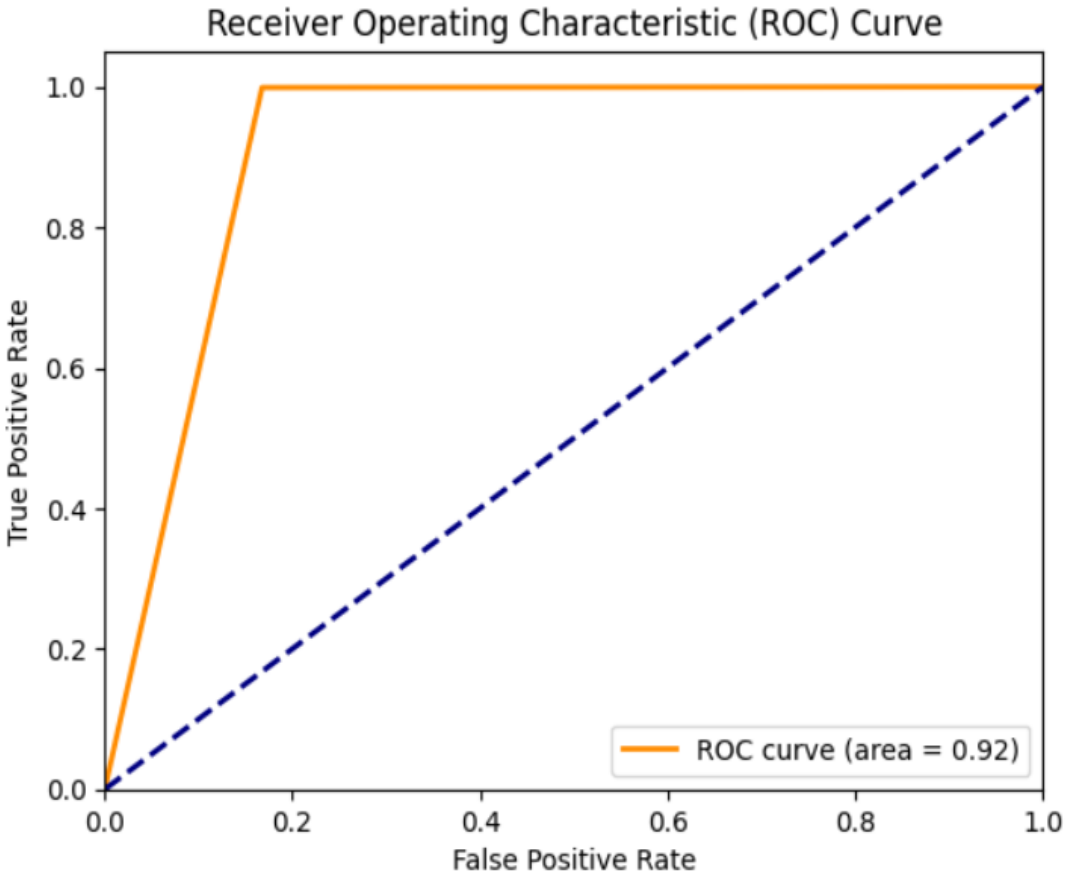
BERT, while slightly lower in accuracy compared to SVM, still achieved a commendable accuracy of 95%. The precision of 100% indicates that the model's positive predictions are highly reliable, with no false positives. However, the recall of 90% suggests that the model may miss some positive instances, impacting its ability to capture all relevant data points. The F1 score of 0.95 reflects a good balance between precision and recall. The ROC AUC score of 0.92 indicates strong discriminatory power but slightly lower than SVM.

EVALUATION METRIC

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	990
1	0.91	1.00	0.95	1010
accuracy			0.95	2000
macro avg	0.95	0.95	0.95	2000
weighted avg	0.95	0.95	0.95	2000

ROC CURVE



Overall Comparision :

Comparing the classification reports of SVM, BERT, and DeBERTa, it is evident that DeBERTa outperforms both SVM and BERT in terms of accuracy, precision, recall, F1 score, and ROC AUC. While SVM and BERT achieved commendable results, DeBERTa's exceptional performance makes it the preferred choice for detecting AI-generated text in essays. The ROC curves further validate DeBERTa's superiority, showing its ability to achieve optimal classification thresholds and discriminate between classes effectively.

In conclusion, DeBERTa emerges as the most accurate, efficient, and precise model for detecting AI-generated text, providing valuable insights for future research and applications in text classification tasks.

9. FUTURE SCOPE

We discuss potential avenues for further research and development based on the findings and limitations of the current study. The chapter outlines future directions to enhance accuracy, expand training data, and address challenges related to AI paraphrasing detection.

Integration of Ensemble Approach:

One avenue for further work involves integrating the SVM, BERT, and DeBERTa models into an ensemble approach. By combining the strengths of multiple models, we aim to enhance classification accuracy and robustness. Ensemble methods, such as stacking or blending, can leverage the diverse capabilities of individual models to achieve superior performance. This integration may involve optimizing model weights, exploring diUerent ensemble architectures, and conducting thorough validation to ensure optimal results.

Incorporation of Llama Model:

The integration of the Llama model represents another promising direction for future work. As a state-of-the-art language model, Llama oUers the potential for higher accuracy and eUiciency in detecting AI-generated text. By incorporating Llama alongside existing models, we can capitalize on its advanced capabilities and further improve the accuracy of our text diUerentiation task. This integration may require fine-tuning Llama on a large dataset and conducting rigorous evaluation to assess its performance in real-world scenarios.

Expansion of Training Data:

Expanding the training data presents an opportunity to enhance the generalization and robustness of our models. By augmenting the dataset with diverse examples of AI-generated and human-written text, we can improve model performance across diUerent domains and language styles. This expansion may involve collecting additional essays from various sources, leveraging data augmentation techniques, and

addressing biases in the training dataset to ensure equitable representation.

Identification of AI Paraphrasing:

Detecting AI paraphrasing represents a specific challenge that warrants further investigation. As AI models become more adept at generating text, they may produce paraphrases that closely resemble human-written content. Future work could focus on developing specialized algorithms or fine-tuning existing models to specifically identify instances of AI paraphrasing. This may involve exploring linguistic features, context-based analysis, and domain-specific knowledge to diUerentiate between original and paraphrased text eUectively.

Evaluation of Real-World Applications:

Finally, further research should involve the evaluation of our models in real-world applications, such as academic integrity monitoring, content moderation, and plagiarism detection. By deploying the models in practical settings and assessing their performance in detecting AIgenerated text, we can validate their eUectiveness and identify areas for improvement. This evaluation may involve collaboration with industry partners, educational institutions, and online platforms to gather feedback and iterate on the models' capabilities. In conclusion, the further work chapter outlines various avenues for advancing research and development in the field of text diUerentiation. By integrating ensemble methods, incorporating advanced language models like Llama, expanding training data, addressing AI paraphrasing, and evaluating real-world applications, we can continue to enhance the accuracy, eUiciency, and applicability of our text diUerentiation task. These eUorts will contribute to the broader goal of ensuring integrity and trustworthiness in textual content across diverse domains and applications.

10. REFERENCES

1. Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

2. Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. arXiv preprint arXiv:2304.04736.

3. Evan Crothers, Nathalie Japkowicz, and Herna L Viktor. 2023. Machine-generated text: A comprehensive survey of threat models and detection methods. IEEE Access.

4. Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. arXiv preprint arXiv:2301.07597.

5. John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. arXiv preprint arXiv:2301.10226.

6. Tharindu Kumarage, Joshua Garland, Amrita Bhattacharjee, Kirill Trapeznikov, Scott Ruston, and Huan Liu. 2023. Stylometric detection of aigenerated text in twitter timelines. arXiv preprint arXiv:2303.03697.

7. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

8. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551.

9. Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. arXiv preprint arXiv:1908.09203
10. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

11. Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pages 142–150.

12. Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. ArXiv, abs/2301.11305.

13. Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium.

14. OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. http://web.archive.org/web/20230109000707/https://openai.com/ blog/chatgpt/. Accessed: 2023-01-10.

15. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-totext transformer. The Journal of Machine Learning Research, 21(1):5485–5551.

16. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

17. Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman ´ Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176bparameter open-access multilingual language model. arXiv preprint arXiv:2211.05100.