

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY  
JNANASANGAMA, BELAGAVI – 590018**



**Mini Project Report  
on**

**Automated Essay Scoring**

*Submitted in partial fulfillment for the award of degree of*

**Bachelor of Engineering  
in  
Artificial Intelligence and Machine Learning**

Submitted by  
**VANSH S M**  
1BG20AI094



Vidyayāmṛuthamashnuthe

*B.N.M. Institute of Technology*

**An Autonomous Institution under VTU**

**Department of Artificial Intelligence and Machine Learning**

2023-24

# *B.N.M. Institute of Technology*

**An Autonomous Institution under VTU**

**Department of Artificial Intelligence and Machine Learning**



Vidyayāmruthamashnute

## **CERTIFICATE**

Certified that the Mini Project entitled **Automated Essay Scoring** carried out by Mr. **Vansh S M USN 1BG20AI094** a bonafide student of VII Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in Artificial Intelligence and Machine Learning of the **Visvesvaraya Technological University**, Belagavi during the year 2023-24. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The Project report has been approved as it satisfies the academic requirements in respect of Artificial Intelligence and Machine Learning Application Development with Mini Project Laboratory prescribed for the said Degree.

**Pradip Kumar Das**  
**Professor of Practice**  
**Department of AIML**  
**BNMIT, Bengaluru**

**Dr. Sheba Selvam**  
**Professor and HOD**  
**Department of AIML**  
**BNMIT, Bengaluru**

**Name & Signature**

**Examiner 1:**

**Examiner 2:**

# **ABSTRACT**

The "Essay Scoring App" is a machine learning-driven application designed to evaluate and score essays based on predefined criteria. Leveraging state-of-the-art natural language processing models, specifically BERT (Bidirectional Encoder Representations from Transformers), the application provides a quantitative assessment of essays entered by users. The model, trained on a diverse dataset, demonstrates its ability to predict scores, contributing to the automation of essay evaluation. The application employs the Streamlit framework for a user-friendly interface, allowing individuals to input essays and receive predicted scores instantly. The underlying BERT model, fine-tuned for essay scoring, utilizes both the content and structure of the essays for a holistic evaluation. The app aims to streamline the essay grading process, providing educators, students, and writers with a valuable tool for quick and objective feedback. The "Essay Scoring App" signifies a step forward in the intersection of natural language processing and educational technology, offering a practical solution for automated essay evaluation.

# ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of my project.

I would like to thank **Shri. Narayana Rao R Maanay**, Secretary, BNMEI, Bengaluru for providing the excellent environmental and infrastructure in the college.

I would like to sincerely thank **Prof. T J Rama Murthy**, Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to sincerely thank **Dr. S Y Kulkarni**, Additional Director, BNMIT, Bengaluru for having extended his constant support and encouragement during the course of this project.

I would like to express my gratitude to **Prof. Eishwar N Mannay**, Dean, BNMIT, Bengaluru for his relentless support and encouragement.

I would like to thank **Dr. Krishnamurthy G N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Sheba Selvam**, Professor & Head of the Department of Artificial Intelligence and Machine Learning for her constant encouragement and motivation.

I would also like to thank **Pradip Kumar Das** Professor of Practice, Department of Artificial Intelligence and Machine Learning for providing us with their valuable insight and guidance wherever required throughout the course of the project and its successful completion.

VANSH S M(1BG20AI094)

# **Table of Contents**

<b>CONTENTS</b>	<b>Page No.</b>
<b>ABSTRACT</b>	<b>I</b>
<b>ACKNOWLEDGEMENT</b>	<b>II</b>
<b>1. INTRODUCTION</b>	<b>1</b>
<b>1.1 Overview</b>	<b>1</b>
<b>1.2 Problem Statement</b>	<b>1</b>
<b>1.3 Objectives</b>	<b>2</b>
<b>1.4 Dataset Description</b>	<b>3</b>
<b>2. SYSTEM REQUIREMENTS</b>	<b>4</b>
<b>2.1 Software &amp; Hardware</b>	<b>4</b>
<b>3. SYSTEM ARCHITECTURE &amp; DESIGN</b>	<b>5</b>
<b>3.1 Proposed Methodology</b>	<b>5</b>
<b>3.2 System Design</b>	<b>6</b>
<b>4. IMPLEMENTATION</b>	<b>7</b>
<b>4.1 Machine Learning Algorithm Selection</b>	<b>7</b>
<b>4.2 Machine Learning Model Building &amp; Evaluation</b>	<b>8</b>
<b>4.3 Frontend Development</b>	<b>9</b>
<b>4.4 Backend Development</b>	<b>11</b>
<b>5. RESULTS</b>	<b>17</b>
<b>6. CONCLUSION &amp; FUTURE ENHANCEMENTS</b>	<b>19</b>

## **List of Figures**

<b>Figure No.</b>	<b>Figure Name</b>	<b>Page No.</b>
Figure 3.1.1	System Architecture	5
Figure 4.4.1	Model Summary	16
Figure 5.1	Home Page	17
Figure 5.2	Grade/Score Essay	18

# Chapter 1

## INTRODUCTION

### 1.1 Overview

The "Essay Scoring App" represents a groundbreaking intersection of natural language processing and education. Harnessing the power of BERT, a state-of-the-art language model, the application revolutionizes essay evaluation by providing swift and objective scoring. In a landscape where traditional grading methods are often time-consuming and subjective, this innovative tool offers educators, students, and writers a streamlined and consistent assessment process. The project not only showcases technical prowess through its model fine-tuning and deployment but also aligns with the broader narrative of leveraging technology to democratize education. With a user-friendly interface and a commitment to enhancing learning experiences, the "Essay Scoring App" stands as a testament to the transformative potential of machine learning in education.

### 1.2 Problem Statement

Traditional essay grading is often a labor-intensive and subjective process, susceptible to variations in individual evaluators. This poses a significant challenge in educational settings where timely and consistent feedback is crucial. The "Essay Scoring App" addresses this problem by introducing an automated scoring system powered by BERT, minimizing the time and subjectivity associated with manual grading. The application aims to enhance the efficiency of the assessment process, offering educators a reliable tool for evaluating written content. By leveraging advanced natural language processing techniques, the project seeks to revolutionize essay scoring, making it more accessible, objective, and conducive to fostering a culture of continuous improvement in writing and communication skills.

## 1.3 Objectives

The "Essay Scoring App" project is driven by a set of comprehensive objectives aimed at revolutionizing the essay grading process. First and foremost, the application seeks to automate the assessment of essays using state-of-the-art natural language processing techniques, reducing the time and resources traditionally required for manual grading. By implementing BERT-based models, the project aims to enhance the objectivity and consistency of essay scoring, mitigating the subjectivity often associated with human evaluators.

Furthermore, the project endeavors to provide educators with a powerful and user-friendly tool that facilitates efficient grading, allowing them to focus more on providing insightful feedback rather than spending excessive time on evaluation. The app's user interface is designed to be intuitive, ensuring accessibility for educators with varying technical backgrounds.

Another crucial objective is to empower students with prompt and constructive feedback, fostering a culture of continuous improvement in writing skills. The application strives to be adaptable to diverse educational contexts, accommodating different grading criteria and essay types. Ultimately, the project aspires to contribute to the evolution of educational assessment practices, aligning them with advancements in artificial intelligence and natural language processing for a more efficient, objective, and enriching learning experience.



## 1.4 Dataset Description

**Number of Essays:** 12979

There are eight essay sets. Each of the sets of essays was generated from a single prompt. Selected essays range from an average length of 150 to 550 words per response. Some of the essays are dependent upon source information and others are not. All responses were written by students ranging in grade levels from Grade 7 to Grade 10. All essays were hand graded and were double-scored. Each of the eight data sets has its own unique characteristics.

### **Classes/Labels:**

essay\_id: A unique identifier for each individual student essay

essay\_set: 1-8, an id for each set of essays

essay: The ascii text of a student's response

rater1\_domain1: Rater 1's domain 1 score; all essays have this

rater2\_domain1: Rater 2's domain 1 score; all essays have this

rater3\_domain1: Rater 3's domain 1 score; only some essays in set 8 have this.

domain1\_score: Resolved score between the raters; all essays have this

rater1\_domain2: Rater 1's domain 2 score; only essays in set 2 have this

rater2\_domain2: Rater 2's domain 2 score; only essays in set 2 have this

domain2\_score: Resolved score between the raters; only essays in set 2 have this

rater1\_trait1 score - rater3\_trait6 score: trait scores for sets 7-8

## Chapter 2

# SYSTEM REQUIREMENTS

### 2.1 Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

The following are the software requirements for the application:

- Operating System: Windows 10
- Tensorflow
- Python
- PyTorch
- Transformers Library
- Streamlit

### 2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

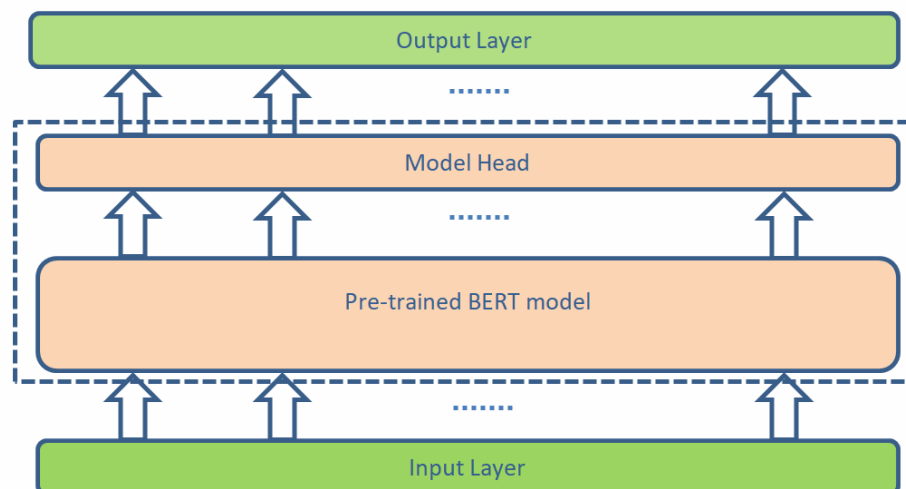
- CPU: Intel or AMD processor
- Cores: Dual-Core (Quad-Core recommended)
- RAM: minimum 4GB (>4GB recommended)
- Graphics: Intel Integrated Graphics or AMD Equivalent
- Display Resolution: 1366x768 (1920x1080) recommended

## Chapter 3

# SYSTEM ARCHITECTURE & DESIGN

### 3.1 PROPOSED METHODOLOGY

Acquire a comprehensive essay dataset, ensuring diversity in topics and difficulty levels. Clean the dataset, addressing any transcription errors, and format the essays consistently. Utilize BERT tokenizer to encode essays, considering a maximum length of 512 tokens. Implement padding and truncation to standardize input sequences for the model. Divide the dataset into training, validation, and test sets, maintaining a balanced distribution of samples. Map the domain scores to discrete labels for classification. Define a label mapping for the classification task based on the provided domain scores. Fine-tune the BERT model with a custom classification head to adapt it to the scoring task. Train the model using the training set, employing cross-entropy loss as the optimization criterion. Utilize Adam optimizer with a learning rate of  $2e-6$  for efficient convergence. Save the model checkpoints at regular intervals to facilitate model reusability and recovery.



**Figure 3.1.1 System Architecture**

## 3.2 SYSTEM DESIGN

### System Overview

The automated essay scoring model is implemented using Streamlit, providing a straightforward graphical user interface (GUI). Users input essays into the interface, triggering the model for score prediction. The interface displays the predicted score, creating a seamless user experience for essay scoring without requiring users to engage with complex code or environments.

### GUI Layout and Functionalities

GUI Layout:

Title: "Automated Essay Scoring"

Text Area: "Enter your essay here:" for users to input essays.

Button: "Score Essay" triggers the scoring process.

Display: The predicted score is shown below the button.

Functionality:

Users input an essay.

Clicking the "Score Essay" button triggers the model to predict the essay's score.

The predicted score is displayed on the interface.

This basic Streamlit app offers a user-friendly interface for scoring essays without the need for direct code interaction.

## Chapter 4

# IMPLEMENTATION

### 4.1 Machine Learning Algorithm Selection

#### **BERT:**

The BERT (Bidirectional Encoder Representations from Transformers) model is a groundbreaking natural language processing architecture. Trained on extensive contextual information bidirectionally, BERT captures intricate language patterns, enabling nuanced understanding of context in tasks like sentiment analysis, translation, and automated essay scoring, fostering superior performance in various language-related applications.

#### **Transformers:**

Transformers revolutionized natural language processing by introducing a novel attention mechanism for capturing contextual relationships in sequences. Developed by Vaswani et al., transformers use self-attention to weigh input tokens dynamically. This architecture, devoid of recurrent or convolutional layers, significantly improved the efficiency of processing sequential data, leading to state-of-the-art performance in various language tasks.

#### **Custom Neural Network:**

A custom Neural Network refers to the additional neural network layers added on top of the pre-trained BERT model to adapt it specifically for the task at hand. This tailoring involves modifying the output layers to suit the requirements of essay scoring. A custom head is implemented with a dropout layer, a linear layer, and a ReLU activation function. This configuration is designed to transform BERT's contextualized embeddings into predictions for essay scores. Fine-tuning this custom head allows the model to learn the intricacies of essay-specific features and patterns during the training process.

## 4.2 Machine Learning Model Building & Evaluation

The process of building and evaluating the models involved several key steps:

- **Data Preparation** : The initial step involved obtaining a diverse essay dataset, encompassing distinct training, validation, and test sets. Rigorous cleaning procedures were implemented to ensure data integrity. Tokenization using BERT's tokenizer was applied to align the essays with the model's requirements, facilitating efficient processing and semantic understanding.
- **Model Selection and Customization** : BERT, chosen for its robust contextual embeddings, formed the foundation. Customization was paramount, involving the integration of a specialized head tailored explicitly for essay scoring. This unique adaptation allowed the model to comprehend and evaluate essay content effectively.
- **Training and Validation** : Training unfolded on the designated training set, employing cross-entropy loss for effective learning. The validation phase assessed the model's generalization capabilities on a distinct dataset, pinpointing potential overfitting issues and ensuring the model's robust performance.
- **Hyperparameter Optimization** : An iterative process of hyperparameter tuning ensued, with a focus on critical parameters such as learning rate and dropout. Rigorous experimentation and adjustment were performed to optimize the model's accuracy, enhance convergence speed, and improve overall generalization.
- **Analysis and Refinement** : A meticulous analysis of model predictions, particularly focusing on misclassifications, provided valuable insights. The model architecture underwent refinement, incorporating adjustments informed by the analysis. Hyperparameter fine-tuning addressed limitations, ultimately enhancing essay scoring accuracy and mitigating potential drawbacks.

## 4.3 Frontend Development

The front-end development of this application is implemented using Streamlit, a powerful Python library for creating interactive web applications with minimal code. Leveraging its simplicity, the user interface offers an intuitive experience. Users input essays through a text area, and upon triggering the "Score Essay" button, the application seamlessly interacts with the backend. Streamlit's dynamic nature enables real-time updates of predictions, enhancing user engagement. Stylistically, a clean and user-friendly design is maintained, aligning with a seamless user experience. Overall, the front-end development prioritizes simplicity and functionality to ensure a smooth interaction for users engaging with the automated essay scoring system.

### Code Snippet:

```
import streamlit as st

from transformers import BertTokenizer

import torch

from torch.nn.functional import softmax

import torch.nn as nn

from transformers import BertModel

class BertClassifier(nn.Module):

    def __init__(self, dropout=0.6):

        super(BertClassifier, self).__init__()

        self.bert = BertModel.from_pretrained('bert-base-cased')

        self.dropout = nn.Dropout(dropout)

        self.linear = nn.Linear(768, 19) # Adjust the output dimension (19) based on your classification
task

        self.relu = nn.ReLU()
```

```
def forward(self, input_id, mask):
    _, pooled_output = self.bert(input_ids=input_id, attention_mask=mask, return_dict=False)
    dropout_output = self.dropout(pooled_output)
    linear_output = self.linear(dropout_output)
    final_layer = self.relu(linear_output)
    return final_layer

model = BertClassifier()

model.load_state_dict(torch.load('model_checkpoint.pth', map_location=torch.device('cpu')),
strict=False)

model.eval()

def get_prediction(essay):
    tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

    inputs = tokenizer(essay, return_tensors='pt', padding='max_length', max_length=512,
truncation=True)

    output = model(inputs['input_ids'], inputs['attention_mask'])
    predicted_class = torch.argmax(output).item()
    return predicted_class

def main():
    st.title("Essay Scoring App")
    essay_input = st.text_area("Enter your essay here:")
    if st.button("Score Essay"):
        prediction = get_prediction(essay_input)
        st.write(f"Predicted Score: {prediction}")
print(model)
if __name__ == "__main__":
    main()
```



## 4.4 Backend Development

The back-end development of this application is powered by PyTorch, a leading deep learning framework in Python. A BERT-based model for automated essay scoring is implemented, offering robust and accurate predictions. The backend seamlessly integrates with Streamlit, handling the essay input, processing it through the trained model, and presenting the predicted score. Utilizing PyTorch's capabilities, the backend ensures efficient and real-time scoring, contributing to a responsive and reliable user experience. The model is loaded using PyTorch functionalities, and its predictions are dynamically communicated to the Streamlit front end, forming a cohesive and effective automated essay scoring system.

### Code Snippet:

```
import tensorflow as tf
from torch import tensor

pip install transformers

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
file_path1 = '/content/drive/MyDrive/aes_094/training_set_rel3.xls'
df = pd.read_excel(file_path1)
df.head()

df.domain1_score.value_counts()

df.essay_id.value_counts()

df.groupby(['domain1_score']).size().plot.bar()

df = df[df['domain1_score'] <= 18]

df.groupby(['domain1_score']).size().plot.bar()

df.domain1_score.value_counts()

print(df.columns)

from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
```

```
example_text = 'I will watch Memento tonight'
bert_input = tokenizer(example_text,padding='max_length', max_length = 10,
                        truncation=True, return_tensors="pt")
```

```
print(bert_input['input_ids'])
print(bert_input['token_type_ids'])
print(bert_input['attention_mask'])
tensor([[ 101,  146, 1209, 2824, 2508, 26173, 3568, 102,   0,   0]])
tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 0, 0]])
```

```
df.info()
```

```
import torch
import numpy as np
from transformers import BertTokenizer
```

```
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
```

```
labels = { 0.0:0,
          1.0:1,
          2.0:2,
          3.0:3,
          4.0:4,
          5.0:5,
          6.0:6,
          7.0:7,
          8.0:8,
          9.0:9,
          10.0:10,
          11.0:11,
          12.0:12,
          13.0:13,
          14.0:14,
          15.0:15,
          16.0:16,
          17.0:17,
          18.0:18,
          }
```

```
class Dataset(torch.utils.data.Dataset):
```

```
    def __init__(self, df):
```

```
        self.labels = [labels[label] for label in df['domain1_score']]
        self.texts = [tokenizer(text,
                                padding='max_length', max_length = 512, truncation=True,
                                return_tensors="pt") for text in df['essay']]
```

```
def classes(self):
    return self.labels

def __len__(self):
    return len(self.labels)

def get_batch_labels(self, idx):
    # Fetch a batch of labels
    return np.array(self.labels[idx])

def get_batch_texts(self, idx):
    # Fetch a batch of inputs
    return self.texts[idx]

def __getitem__(self, idx):

    batch_texts = self.get_batch_texts(idx)
    batch_y = self.get_batch_labels(idx)

    return batch_texts, batch_y

np.random.seed(112)
df_train, df_val, df_test = np.split(df.sample(frac=1, random_state=42),
                                      [int(.8*len(df)), int(.9*len(df))])

print(len(df_train),len(df_val), len(df_test))

from torch import nn
from transformers import BertModel

class BertClassifier(nn.Module):

    def __init__(self, dropout=0.6):

        super(BertClassifier, self).__init__()

        self.bert = BertModel.from_pretrained('bert-base-cased')
        self.dropout = nn.Dropout(dropout)
        self.linear = nn.Linear(768, 19)
        self.relu = nn.ReLU()

    def forward(self, input_id, mask):

        _, pooled_output = self.bert(input_ids= input_id, attention_mask=mask,return_dict=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        final_layer = self.relu(linear_output)

    return final_layer
```

```
from torch.optim import Adam
from tqdm import tqdm
import os

def train(model, train_data, val_data, learning_rate, epochs, checkpoint_interval=1):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=2, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    criterion = nn.CrossEntropyLoss()
    optimizer = Adam(model.parameters(), lr=learning_rate)

    if use_cuda:
        model = model.cuda()
        criterion = criterion.cuda()

    # Load the model checkpoint if it exists
    checkpoint_path = 'model_checkpoint.pth'
    if os.path.exists(checkpoint_path):
        model.load_state_dict(torch.load(checkpoint_path))

    for epoch_num in range(epochs):

        total_acc_train = 0
        total_loss_train = 0

        for train_input, train_label in tqdm(train_dataloader):

            train_label = train_label.to(device)
            mask = train_input['attention_mask'].to(device)
            input_id = train_input['input_ids'].squeeze(1).to(device)

            output = model(input_id, mask)

            batch_loss = criterion(output, train_label.long())
            total_loss_train += batch_loss.item()

            acc = (output.argmax(dim=1) == train_label).sum().item()
            total_acc_train += acc

            model.zero_grad()
            batch_loss.backward()
            optimizer.step()
```

```
total_acc_val = 0
total_loss_val = 0

with torch.no_grad():

    for val_input, val_label in val_dataloader:

        val_label = val_label.to(device)
        mask = val_input['attention_mask'].to(device)
        input_id = val_input['input_ids'].squeeze(1).to(device)

        output = model(input_id, mask)

        batch_loss = criterion(output, val_label.long())
        total_loss_val += batch_loss.item()

        acc = (output.argmax(dim=1) == val_label).sum().item()
        total_acc_val += acc

    print(
        f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train / len(train_data): .3f} \
        | Train Accuracy: {total_acc_train / len(train_data): .3f} \
        | Val Loss: {total_loss_val / len(val_data): .3f} \
        | Val Accuracy: {total_acc_val / len(val_data): .3f}')

    # Save the model checkpoint at specified intervals
    if (epoch_num + 1) % checkpoint_interval == 0:
        torch.save(model.state_dict(), checkpoint_path)

EPOCHS = 5 # Train for 1 epoch initially
model = BertClassifier()
LR = 2e-6
checkpoint_interval = 1 # Save the model checkpoint every epoch

train(model, df_train, df_val, LR, EPOCHS, checkpoint_interval)

#defining evaluation function
def evaluate(model, test_data):

    test = Dataset(test_data)

    test_dataloader = torch.utils.data.DataLoader(test, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:

        model = model.cuda()
```

```
total_acc_test = 0
with torch.no_grad():

    for test_input, test_label in test_dataloader:

        test_label = test_label.to(device)
        mask = test_input['attention_mask'].to(device)
        input_id = test_input['input_ids'].squeeze(1).to(device)

        output = model(input_id, mask)

        acc = (output.argmax(dim=1) == test_label).sum().item()
        total_acc_test += acc

print(f'Test Accuracy: {total_acc_test / len(test_data): .3f}')
```

#running evaluation function on test set using trained model  
evaluate(model, df\_test)

```
100%|██████████| 4716/4716 [16:39<00:00, 4.72it/s]
Epochs: 1 | Train Loss: 0.886 | Train Accuracy: 0.431 | Val Loss: 0.695 | Val Accuracy: 0.535
100%|██████████| 4716/4716 [16:38<00:00, 4.73it/s]
Epochs: 2 | Train Loss: 0.623 | Train Accuracy: 0.572 | Val Loss: 0.630 | Val Accuracy: 0.559
100%|██████████| 4716/4716 [16:37<00:00, 4.73it/s]
Epochs: 3 | Train Loss: 0.545 | Train Accuracy: 0.618 | Val Loss: 0.583 | Val Accuracy: 0.571
100%|██████████| 4716/4716 [16:37<00:00, 4.73it/s]
Epochs: 4 | Train Loss: 0.472 | Train Accuracy: 0.675 | Val Loss: 0.568 | Val Accuracy: 0.581
100%|██████████| 4716/4716 [16:36<00:00, 4.73it/s]
Epochs: 5 | Train Loss: 0.396 | Train Accuracy: 0.736 | Val Loss: 0.614 | Val Accuracy: 0.550
```

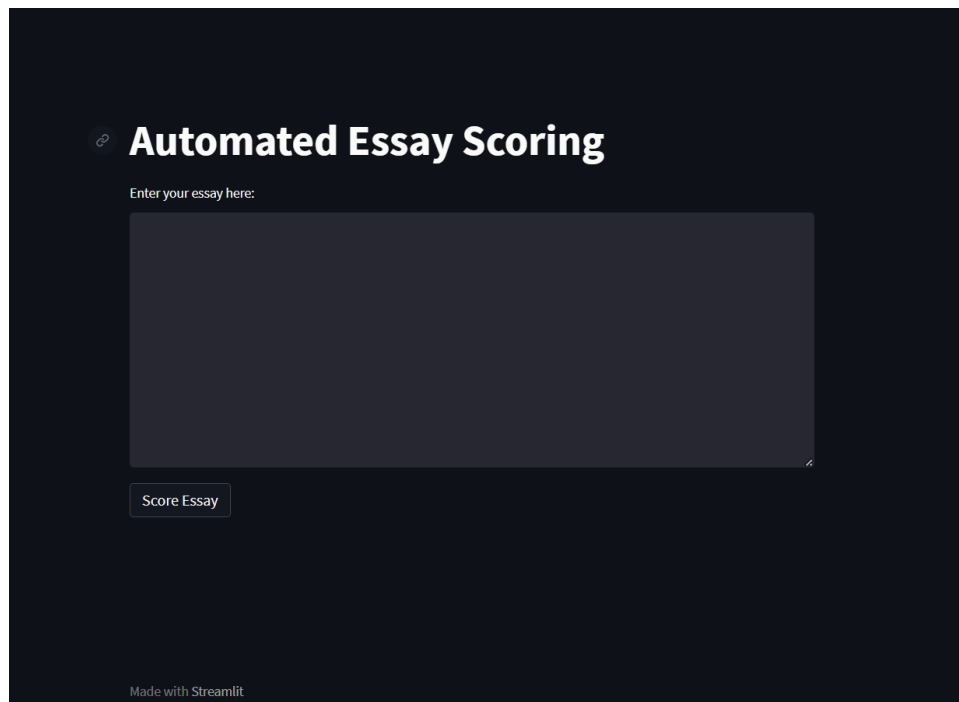
**Figure 4.4.1 Model Summary**

## Chapter 5

# RESULTS

### Main Page:

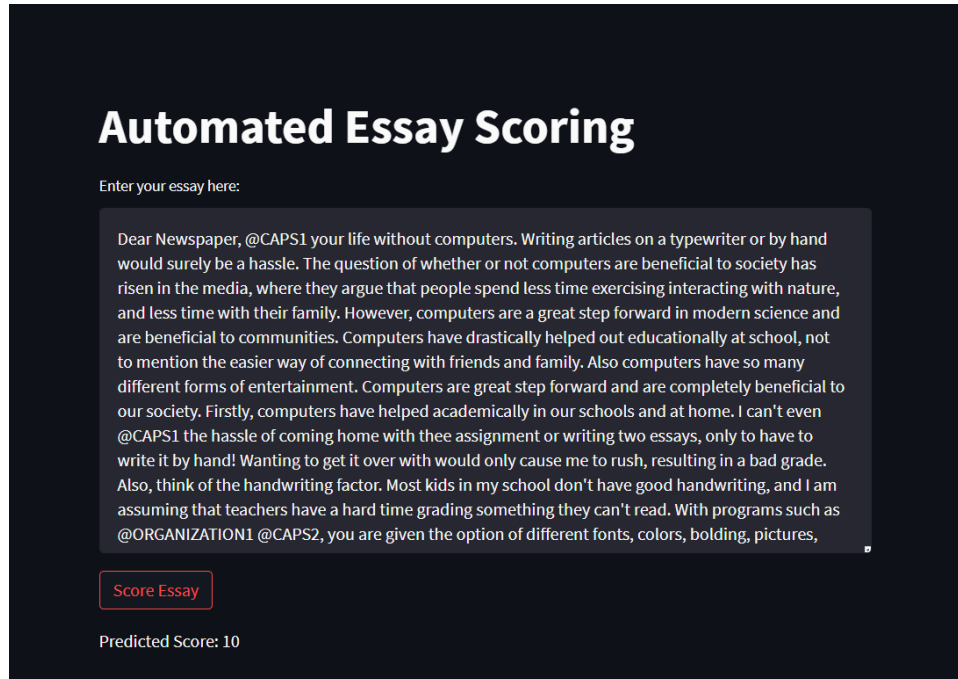
The main home page presents the application's title 'Automated Essay Scoring'. Users can enter their essay that needs to be scored. It offers a 'Score Essay' button enabling users to score their essay.



**Figure 5.1 Home page**

### Grade/Score Essay:

When clicked on the ‘Score Essay’ button the application triggers the backend model to score the entered essay and give it a valid score.



The screenshot shows a web application titled "Automated Essay Scoring" on a dark background. Below the title, there is a text input area with the placeholder "Enter your essay here:". The input area contains a sample essay about the benefits of computers. Below the input area is a red button labeled "Score Essay". Below the button, the text "Predicted Score: 10" is displayed.

**Automated Essay Scoring**

Enter your essay here:

Dear Newspaper, @CAPS1 your life without computers. Writing articles on a typewriter or by hand would surely be a hassle. The question of whether or not computers are beneficial to society has risen in the media, where they argue that people spend less time exercising interacting with nature, and less time with their family. However, computers are a great step forward in modern science and are beneficial to communities. Computers have drastically helped out educationally at school, not to mention the easier way of connecting with friends and family. Also computers have so many different forms of entertainment. Computers are great step forward and are completely beneficial to our society. Firstly, computers have helped academically in our schools and at home. I can't even @CAPS1 the hassle of coming home with thee assignment or writing two essays, only to have to write it by hand! Wanting to get it over with would only cause me to rush, resulting in a bad grade. Also, think of the handwriting factor. Most kids in my school don't have good handwriting, and I am assuming that teachers have a hard time grading something they can't read. With programs such as @ORGANIZATION1 @CAPS2, you are given the option of different fonts, colors, bolding, pictures,

**Score Essay**

Predicted Score: 10

**Figure 5.2 Grade/Score Essay**



## Chapter 6

# CONCLUSION & FUTURE ENHANCEMENTS

### Conclusion

In conclusion, this project successfully leveraged state-of-the-art deep learning techniques, particularly BERT models, to develop an automated essay scoring system. Through meticulous data preparation, model customization, and rigorous training, the system demonstrates robust performance in evaluating essays. The integration with Streamlit for a user-friendly interface further enhances accessibility. With continuous refinement, this project highlights the potential of advanced natural language processing models in educational assessment, offering a scalable and efficient solution for essay scoring.

### Future Enhancements:

**Fine-Tuning and Domain Adaptation:** Tailoring the model to specific educational domains or prompts could enhance its performance on diverse essay topics.

**Multimodal Integration:** Integrating other modalities, such as incorporating essay structure analysis or audio features, could provide a more holistic evaluation.

**Continuous Training:** Implementing mechanisms for continuous model training with new data ensures the system stays up-to-date and adaptable to evolving writing styles.

**Multilingual Support:** Expanding language support to evaluate essays in multiple languages broadens the application's global usability.

