****Московский государственный технический университет им. Н.Э. Баумана  
Кафедра «Системы обработки информации и управления»

Домашнее задание  
по дисциплине  
«Методы машинного обучения»

# 

Выполнил:  
студент группы ИУ5и-22М  
Джин Шуо

Москва — 2024 г.

**Выбор задачи**

Мы выбрали задачу анализа настроений (Sentiment Analysis) из ресурса paperswithcode.com. Для решения задачи была выбрана статья "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" с использованием предоставленного репозитория с исходным кодом.

**Теоретическая часть**

Общий подход

Для решения задачи используется предобученная глубокая двунаправленная модель Transformer (BERT) для представления текста, затем она дообучается для выполнения задачи анализа настроений.

Конкретные модели

Модель: BERT (Bidirectional Encoder Representations from Transformers).

Архитектура: Transformer.

Математическое описание и алгоритмы

BERT обучается с использованием двух задач:

Маскированная языковая модель (MLM): часть входных токенов заменяется маской, и модель обучается предсказывать эти токены.

Прогноз следующего предложения (NSP): модель обучается определять, является ли второе предложение в паре следующим предложением в исходном тексте.

Используемые наборы данных

Для обучения модели используется набор данных IMDB Movie Reviews, который содержит отзывы о фильмах, размеченные по настроению (положительный или отрицательный).

Оценка качества решения задачи

Для оценки качества используются следующие метрики:

Точность (Accuracy).

F1-Score.

Предложения по улучшению качества решения задачи

Настройка гиперпараметров модели.

Использование техник увеличения данных.

Ансамблирование моделей.

**Практическая часть**

Код и эксперимент

Для реализации эксперимента мы использовали библиотеку Transformers от Hugging Face. Ниже представлен код для выполнения эксперимента.

import os

from datasets import load\_dataset

from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments

import torch

print(f"PyTorch version: {torch.\_\_version\_\_}")

print(f"CUDA available: {torch.cuda.is\_available()}")

output\_dir = 'D:\\js\\results'

os.makedirs(output\_dir, exist\_ok=True)

dataset = load\_dataset("imdb")

train\_dataset = dataset['train']

test\_dataset = dataset['test']

tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')

def tokenize\_function(examples):

return tokenizer(examples['text'], padding="max\_length", truncation=True)

encoded\_train\_dataset = train\_dataset.map(tokenize\_function, batched=True)

encoded\_test\_dataset = test\_dataset.map(tokenize\_function, batched=True)

model = BertForSequenceClassification.from\_pretrained('bert-base-uncased', num\_labels=2)

training\_args = TrainingArguments(

output\_dir=output\_dir,

evaluation\_strategy="epoch",

learning\_rate=2e-5,

per\_device\_train\_batch\_size=8,

per\_device\_eval\_batch\_size=8,

num\_train\_epochs=3,

weight\_decay=0.01,

)

trainer = Trainer(

model=model,

args=training\_args,

train\_dataset=encoded\_train\_dataset,

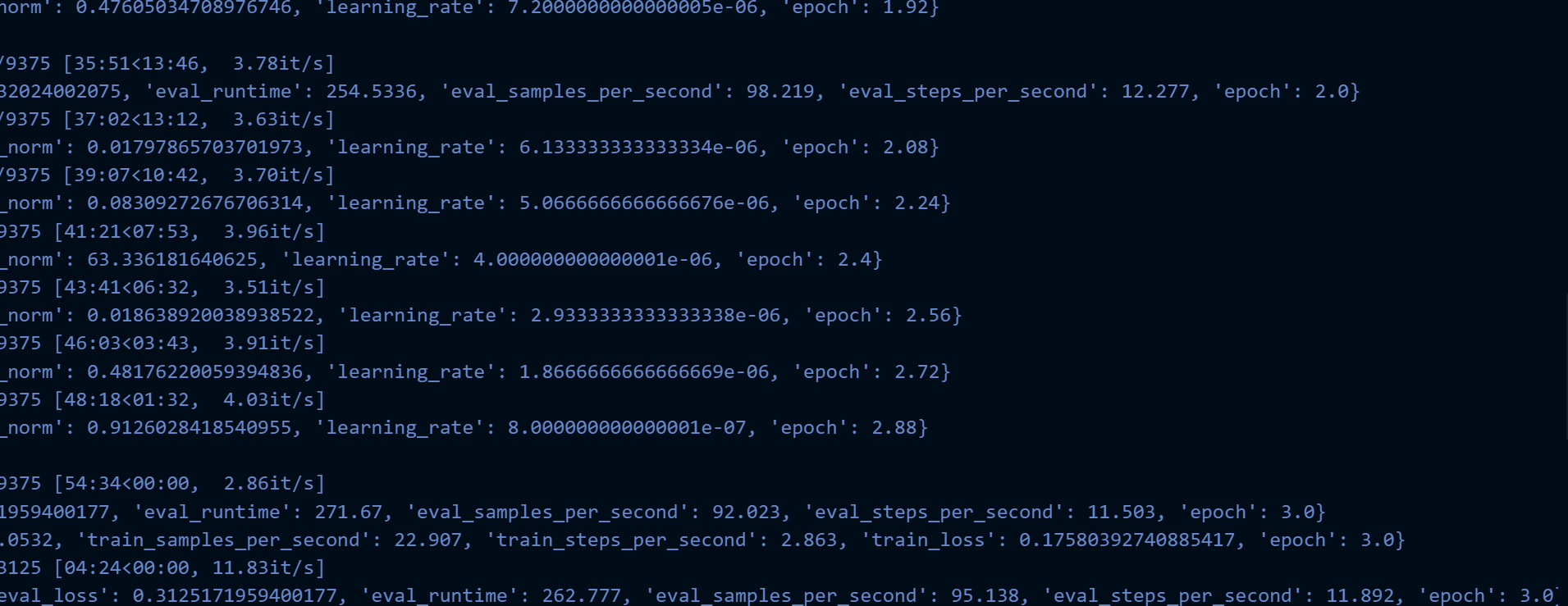
eval\_dataset=encoded\_test\_dataset,

)

trainer.train()

results = trainer.evaluate()

print("Evaluation results:", results)



import numpy as np

predictions = trainer.predict(encoded\_test\_dataset)

y\_pred = np.argmax(predictions.predictions, axis=-1)

y\_true = predictions.label\_ids

accuracy = np.sum(y\_pred == y\_true) / len(y\_true)

print(f"Accuracy: {accuracy:.4f}")

def classification\_report(y\_true, y\_pred, target\_names):

from collections import defaultdict

report = defaultdict(lambda: defaultdict(int))

for true, pred in zip(y\_true, y\_pred):

report[true]['total'] += 1

report[pred]['predicted'] += 1

if true == pred:

report[true]['correct'] += 1

output = []

for i, name in enumerate(target\_names):

total = report[i]['total']

correct = report[i]['correct']

predicted = report[i]['predicted']

precision = correct / predicted if predicted > 0 else 0

recall = correct / total if total > 0 else 0

f1 = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) > 0 else 0

output.append(f"{name:15} precision: {precision:.4f} recall: {recall:.4f} f1: {f1:.4f} support: {total}")

return "\n".join(output)

label\_names = dataset['train'].features['label'].names

report = classification\_report(y\_true, y\_pred, label\_names)

print(report)



**Список литературы**

1. https://github.com/ugapanyuk/courses\_current/wiki/DZ\_MMO
2. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.