LLM Assignment#1

Q1:

Project Idea:

I like to choose the project that associates with SDGs sustainable Development Goals. Hence, the project I have chosen is focused on quality education "SDG 4" by developing the model that assist in automated marking or grading of offering personalized learning or essays recommendations. Hence, automated grading systems (AES) is known for automated essay scoring also which are software instruments use AI artificial intelligence for grading and assessing student work. The systems analyse numerous kinds of assignment including essays, coding projects, and multiple choice questions (Garcia, 2023).

Detailed explanation:

In the project of automated grading, our goal is creating the system that can score and evaluate essays according to various criteria like grammar, relevance, and coherence to the subject. Additionally, the objective is developing the model that can examine the pattern of students learning and providing tailored suggestions to promote their experience in learning. Thus, this can aid educators provide more reliable and consistent feedback, reduce workload for instructor, and also save time to students (Garcia, 2023).

Part One: Research and Setup:

First, we investigate the Library of Hugging Face Transformers:

The library of hugging face transformers is referred to the powerful instrument for NLP natural language processing tasks. This tool can provide pre-trained models to tasks like question answering, T5, text generation, classification, and so on.

Model Selection:

In this part, we browse the hub of hugging face model for finding an appropriate pre-trained model for our task. For our project (automated essay grading), we consider models such as GPT 3, RoBERT, and BERT, which are the well-suited tool for generating and understanding text. Bert for essay grading correctly interpret the meaning of the sentences, evaluate the relevance of the content to the topic, and understand complex grammar. Also, RoBERTa

enhanced performance makes particularly it valued for more reliable and accurate grading, specifically in handling varied writing styles and language patterns.

Part Two: Implementation

Here we can load the pre-trained model by using the library of hugging face. We load the model in Python.

From transformers, we import AutoModelForSequenceClassification, AutoTokenizer

Model_name= “bert-base-uncased” model= AutoModelForSequenceClassification.from_pretrained (model_name)tokenizer=AutoTokenizer.from_ pretrained(model_name)

Data Preparation:

Here I used dataset of ASAP Automated Essay Scoring

Perform EDA:

EDA is for better understanding the dataset and recognizing any preprocessing steps that is required.

Test the Model Before Fine-Tuning:

Here we test the pretrained model on the small subset of our data for establishing the baseline performance.

Fine Tuning the model and preprocess:

Here we preprocess our data for matching the input requirements of the model. We fine tune the model on our dataset.

From transformers, we import Trainer, trainingarguments
Training_args=trainingarguments(output_dir= “./results”, evaluation-strategy= “epoch”, learning_rate=2e-5,per_device_train_batch-size=16,per_device_eval_batch_size=16,num_train_epochs=3,weight_decay=0.01,)
Trainer=trainer (model=model,args=traning +args,train_dataset=train_dataset,eval_dataset=eval_dataset,)

Trainer.train()

Part three Evaluation:

Here we compare and evaluate the performance of the model by using metrics like F1 score, BLUE score, and accuracy for evaluating the performances of the model before and after fine tuning. Also, we compare the results for determining the effectiveness of our fine tuning process.

Q2:

Fine Tuning Large Language Models for QD quality education SDG4

Project idea:

Fine tune a (LLM) large language model to enhance its performance on exact educational tasks like personalized learning recommendations or automated essay grading, aligned with Quality Education SDG4.

Project Explanation:

The goal of the project is improving the capabilities of the pre trained large language model (LLM) for better handling tasks that are interrelated to education. Via fine tuning a model, we pursue to get more context specific and accurate results, which contributes to enhance educational outcomes ultimately.

Part one: Setup and Research:

Here we install the library of hugging face transformers first.

Pip install transformers datasets

Secondly, we explore EDA data analysis. For this project (essay grading) we use the dataset of ASAP automated essay scoring.

Third, we evaluate the dataset to recognize its characteristics and structure.

Here is the code:

We import pd for pandas

Dt=pd.read_csv (‘path_to_dataset.csv’)print(Dt.head())

We also use visualization instruments to get perceptions into the numbers or data.

We import plt instead of matplotlib.pyplot, and sns for seaborn

sns.countplot(Dt[‘score’]) plt.show()

Part Two:

We preprocess the numeric or data to make appropriate for the certain task. For instance, tokenization for manuscript or text data.

We import AutoTokenizer from transformers
Tokenizer=Autokenizer.from_pretrained (‘bert-base-uncased’)
Dt[‘tokens’]=Dt[‘essay’].apply(lambdax:tokenizer.encode(x, truncation=True,padding= ‘max_length’))

Part Three:

We chose a pretrianed model from the hub of hugging face model.
From transformers, we import AutoModelForSequenceClassification
Model=AutoForSequenceClassification.from *pretrained (‘bert-base-uncaseed’,num*_labels=5)

Part Fourth:

Here we define the training arguments, we set up the parameters of training.

We import trainingArguments from transformers.
Training_args=trainingArguments(output_dir= ‘./results’, evaluation_strategy= ‘epoch’,learning_rate=2e-5,per_device_train_batch_size=16,per_device_eval_batch_size=16,num_train_epochs=3,weight_decay=0.01,)

Then we fine tune the model on the dataset that are prepared.

We import trainer from the transformers.
Trainer=trainer(model=model,
args=training_args,train_dataset=train_dataset,eval_dataset=eval_dataset,)
Trainer.train()

Part Fifth, Evaluation:

Here we evaluate the performance of model on the test set and then compare it after and before fine-tuning.

Results=trainer.evaluate ()

Print (results)

Thus, by following these stages, we can fine tune the LLM large Language Model for enhancing its performances on certain educational tasks that contribute to the objective of QE quality education.

References

Garcia, J. August 30, 2023. Streamline Your Grading Process with Automated Grading Systems. Improvitz Formerly Impactum. Retrieved on August 11, 2024 from https://impactum.mx/streamline-your-grading-process-with-automated-grading-systems/