## Introduction

This document outlines the process of fine-tuning the BERT model for sequence classification on the Quora dataset. The goal of this research is to classify pairs of questions as either duplicates or not duplicates. The implementation is done using the Hugging Face transformers library and PyTorch.

## 1. Environment Setup

### 1.1 Import Required Libraries

The necessary libraries for this project include the transformers library for model and tokenizer, torch for tensor operations, and datasets for handling data.

import transformers

import torch

from transformers import AdamW, AutoTokenizer, AutoModelForSequenceClassification, Trainer

from transformers import TrainingArguments

from datasets import ClassLabel, Value

from transformers import DataCollatorWithPadding

import numpy as np

from datasets import load\_metric

### 1.2 Define the Checkpoint

I use the bert-base-uncased checkpoint for this sequence classification task.

checkpoint = "bert-base-uncased"

### 1.3 Initialize the Tokenizer and Model

The BERT tokenizer and sequence classification model are loaded from the checkpoint.

tokenizer = AutoTokenizer.from\_pretrained(checkpoint)

model = AutoModelForSequenceClassification.from\_pretrained(checkpoint, num\_labels=2)

### 1.4 Set Device

I check for the availability of a GPU (CUDA), otherwise default to the CPU.

device = torch.device("cuda") if torch.cuda.is\_available() else torch.device("cpu")

model.to(device)

## 2. Dataset Preparation

### 2.1 Load the Dataset

I load the Quora dataset, which contains pairs of questions and their labels.

from datasets import load\_dataset

raw\_datasets = load\_dataset("quora")

### 2.2 Explore the Dataset

Explore the structure and contents of the dataset.

raw\_datasets['train'].features

raw\_datasets['train'][0:2]

### 2.3 Tokenize the Dataset

I define a tokenization function to process pairs of questions. The tokenizer processes two inputs simultaneously (question pairs).

def tokenize\_function(example):

questions = example['questions']

t1 = []

t2 = []

for t in questions:

t1.append(t['text'][0])

t2.append(t['text'][1])

return tokenizer(t1, t2, truncation=True)

tokenized\_datasets = raw\_datasets['train'].map(tokenize\_function, batched=True)

### 2.4 Modify Dataset Features

I modify the dataset's features to include class labels for 'duplicate' and 'not\_duplicate', remove unnecessary columns, and split the dataset into training and testing sets.

new\_features = tokenized\_datasets.features.copy()

new\_features["is\_duplicate"] = ClassLabel(num\_classes=2, names=['not\_duplicate', 'duplicate'])

tokenized\_datasets = tokenized\_datasets.cast(new\_features)

tokenized\_datasets = tokenized\_datasets.remove\_columns('questions').rename\_column('is\_duplicate', 'labels')

tokenized\_datasets = tokenized\_datasets.train\_test\_split(test\_size=0.2)

## 3. Model Training

### 3.1 Data Collation

Prepare the data for model training with padding.

data\_collator = DataCollatorWithPadding(tokenizer=tokenizer)

### 3.2 Define Metric

I define a function to compute metrics, specifically the MRPC (Microsoft Research Paraphrase Corpus) metrics.

def compute\_metrics(eval\_preds):

metric = load\_metric("glue", "mrpc")

logits, labels = eval\_preds

predictions = np.argmax(logits, axis=-1)

return metric.compute(predictions=predictions, references=labels)

### 3.3 Define Training Arguments

Specify training arguments, such as the number of epochs, batch size, evaluation strategy, and saving strategy.

training\_args = TrainingArguments(

"./quora-saved-model",

evaluation\_strategy="epoch",

save\_strategy='no',

report\_to='none',

num\_train\_epochs=3,

per\_device\_train\_batch\_size=32,

per\_device\_eval\_batch\_size=32

)

### 3.4 Initialize Trainer and Train the Model

Use the Trainer API to initialize the model training.

trainer = Trainer(

model,

training\_args,

train\_dataset=tokenized\_datasets['train'],

eval\_dataset=tokenized\_datasets['test'],

data\_collator=data\_collator,

tokenizer=tokenizer,

compute\_metrics=compute\_metrics,

)

trainer.train()

## 4. Prediction on New Samples

### 4.1 Tokenize Sample Data

Tokenize new pairs of questions for prediction.

tokens = tokenizer([

['How can I be successful in Kaggle Competitions?', 'How can I be successful in life?'],

['What is the best place to eat a pizza in Italy?','What is the best restaurant in Italy?'],

['What are the good courses to learn PyTorch?','Are there good courses to learn PyTorch?']

], truncation=True, padding=True, return\_tensors='pt')

tokens.to(device)

### 4.2 Make Predictions

Use the trained model to make predictions on the new tokenized data.

logits = model(\*\*tokens).logits

logits = logits.cpu().detach().numpy()

preds = np.argmax(logits, axis=-1)

preds

## 5. Conclusion

This document presents the complete process of fine-tuning the BERT model for sequence classification on the Quora dataset. The research demonstrates the ability to classify pairs of questions as either duplicates or not duplicates using a pre-trained transformer model. Further evaluation and hyperparameter tuning could be done to enhance the model's performance.