Assignment 5: Transformers and Natural Language Processing (Part 2, V2)

Note: there is a separate submission portal for part 2 on Moodle

Aspen Morgan

Netid: 790907699

Note: this assignment falls under collaboration Mode 2: Individual Assignment – Collaboration Permitted. Please refer to the syllabus for additional information.

Problem 3: Text Classification with A Large Language Model (30 points) In this example you will utilize a modern large language model to classify text. Specifically, you will use load the pre-trained BERT encoder that we discussed in class, and then fine-tune it to solve a custom text classification problem where you classify news articles into one of four categories: world, sports, business, sci/tech.

To assist with this exercise, we will need to make use of some libraries from Hugging Face, an organization that provides many widely-used libraries to support deep learning applications (link).

Below is a code skeleton for completing this task, with comments to guide you through the process of completing it. Please complete the code below and submit a pdf of your completed code with results. *There will be a separate submission portal for this question on Moodle. Although your code will be reviewed, you will be graded primarily based upon the correctness of your output*

Although the code skeleton below provides useful guidance/hints to fill in teh code, I highly recommend that you review a tutorial on text classification provided by hugging face before, or while, you complete this exercise (tutorial link)

Installations: Make sure you use pip or conda to install the following libraries for this exercise: datasets, evaluate, metrics, transformers, numpy, and torch.

Google Colab already has torch and numpy, but you will still need to install transformers, datasets, evaluate and metrics. You can copy and paste the line below into colab and it will install them.

pip install transformers datasets evaluate accelerate

```
pip install transformers datasets evaluate accelerate
```

```
# Necessary Imports
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer
import numpy as np
import torch

# Load the AG News dataset using load_dataset
dataset = load_dataset("ag_news")
train_dataset = dataset['train']
test_dataset = dataset['test']

# Load the tokenizer for a BERT-based model "TinyBERT", and specify the number of labels
tokenizer = AutoTokenizer.from_pretrained("huawei-noah/TinyBERT_General_4L_312D",num_labels=4)

# Define a function to tokenize the data
def tokenize_function(examples):
    return tokenizer(examples['text'], padding="max_length", truncation=True, max_length=128)

# Tokenize the training and testing data. Hint: use .map to apply the tokenize function above to your train and test datase
train_dataset = train_dataset.map(tokenize_function)
test_dataset = test_dataset.map(tokenize_function)

# Load TinyBERT model We use TinyBERT, which requires substantially less
# compute than BERT, with only a modest reduction in accuracy
model = AutoModelForSequenceClassification.from_pretrained("huawei-noah/TinyBERT_General_4L_312D", num_labels=4)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Downloading readme: 100%                8.07k/8.07k [00:00<00:00, 128kB/s]
Downloading data: 100%                  18.6M/18.6M [00:00<00:00, 25.8MB/s]
Downloading data: 100%                  1.23M/1.23M [00:00<00:00, 10.4MB/s]
Generating train split: 100%            120000/120000 [00:00<00:00, 367823.66 examples/s]
Generating test split: 100%             7600/7600 [00:00<00:00, 145458.12 examples/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is depre
  warnings.warn(
config.json: 100%                       409/409 [00:00<00:00, 11.4kB/s]
vocab.txt: 100%                         232k/232k [00:00<00:00, 3.03MB/s]
Map: 100%                               120000/120000 [00:57<00:00, 2595.10 examples/s]
Map: 100%                               7600/7600 [00:04<00:00, 2177.06 examples/s]
pytorch_model.bin: 100%                 62.7M/62.7M [00:00<00:00, 127MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at huawei-noah/TinyBERT_Ger
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```python
# Define the training arguments
training_args = TrainingArguments(
    output_dir='./results',              # output directory
    num_train_epochs=1,                  # number of training epochs (switched to 1)
    per_device_train_batch_size=8,       # batch size for training
    per_device_eval_batch_size=16,       # batch size for evaluation
    warmup_steps=500,                    # number of warmup steps for learning rate scheduler
    weight_decay=0.01,                   # strength of weight decay
    logging_dir='./logs',                # directory for storing logs
    logging_steps=100,
    evaluation_strategy="epoch"
)


# Function to compute accuracy of the model
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=1)
    return {'accuracy': (predictions == labels).mean()}


# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)

# Train the model
trainer.train()
```

[15000/15000 07:08, Epoch 1/1]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.247900	0.264412	0.928026

```
TrainOutput(global_step=15000, training_loss=0.3376785260518392, metrics={'train_runtime': 430.5741,
'train_samples_per_second': 278.698, 'train_steps_per_second': 34.837, 'total_flos': 430227578880000.0, 'train_loss':
0.3376785260518392, 'epoch': 1.0})
```

```python
# Evaluate the model
results = model.eval()
print(results)
```

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 312, padding_idx=0)
      (position_embeddings): Embedding(512, 312)
      (token_type_embeddings): Embedding(2, 312)
      (LayerNorm): LayerNorm((312,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-3): 4 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=312, out_features=312, bias=True)
              (key): Linear(in_features=312, out_features=312, bias=True)
              (value): Linear(in_features=312, out_features=312, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=312, out_features=312, bias=True)
              (LayerNorm): LayerNorm((312,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=312, out_features=1200, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=1200, out_features=312, bias=True)
            (LayerNorm): LayerNorm((312,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=312, out_features=312, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=312, out_features=4, bias=True)
)
```

```
num_examples = 6

def get_example(data, idx):
    return data['text'][idx], data['label'][idx]

# Make a label mapping dictionary for the AG News dataset (keys should be numbers and values should be the category as a stri
label_map = {0: 'World', 1: 'Sports', 2: 'Business', 3: 'Sci/Tech'}

# Select num_examples examples from the test dataset
examples_text = []
examples_label = []
for i in range(num_examples):
    text, label = get_example(test_dataset, i)
    examples_text.append(text)
    examples_label.append(label)

# TODO: Tokenize the examples
# Hint: similar to how we defined the tokenize_function above, except here you also want to set return_tensors="pt"
# to ensure that the output from the tokenizer is ready for a PyTorch model
inputs = [tokenizer(text, padding="max_length", truncation=True, max_length=128, return_tensors="pt") for text in examples_te


# Move to the same device as model
if torch.cuda.is_available():
    inputs = [{k: v.cuda() for k, v in inp.items()} for inp in inputs]
    model.cuda()

# # For people with a GPU on a Macintosh machine, uncomment this
# elif torch.backends.mps.is_available():
#     inputs = [input.to(device) for input in inputs]
#     device = torch.device("mps")
#     model = model.to(device)
#     model = model.to(device)


# Get predictions
with torch.no_grad():
    outputs = [model(**inp) for inp in inputs]

# Extract logits from the output and apply softmax to get probabilities
# Hint: ModelOutput class documentation https://huggingface.co/docs/transformers/en/main_classes/output
probabilities = [output.logits for output in outputs]

# Get the predicted class indices
predicted_classes = [torch.argmax(prob, dim=-1) for prob in probabilities]

# Print 6 examples where you have the example text on one line, and the true and predicted labels on the next
for i in range(num_examples):
    text, label = get_example(test_dataset, i)
    pred = predicted_classes[i][0]
    print(text)
    print(f'Actual label is {label} and predicted label is {pred}')
    print()
```

```
Fears for T N pension after talks Unions representing workers at Turner   Newall say they are 'disappointed' after talk
Actual label is 2 and predicted label is 2

The Race is On: Second Private Team Sets Launch Date for Human Spaceflight (SPACE.com) SPACE.com - TORONTO, Canada -- A
Actual label is 3 and predicted label is 3

Ky. Company Wins Grant to Study Peptides (AP) AP - A company founded by a chemistry researcher at the University of Lou
Actual label is 3 and predicted label is 3

Prediction Unit Helps Forecast Wildfires (AP) AP - It's barely dawn when Mike Fitzpatrick starts his shift with a blur
Actual label is 3 and predicted label is 3

Calif. Aims to Limit Farm-Related Smog (AP) AP - Southern California's smog-fighting agency went after emissions of the
Actual label is 3 and predicted label is 3

Open Letter Against British Copyright Indoctrination in Schools The British Department for Education and Skills (DfES)
Actual label is 3 and predicted label is 3
```