fine-tunining-model

October 27, 2024

1. Created own dataset for text classification. It should contain at least 2000 words in total and at least three categories with at least 100 examples per category (an example can be a poem or a paragraph from a book).

[1]:
```
import pandas as pd
import numpy as np

file_path = '/content/Labeled_Unique_Text_Classification_Dataset.txt'
data = pd.read_csv(file_path, delimiter=':', header=None, names=['Category',␣
  ↪'Text'])
data.head()
```

[1]:
```
     Category                                               Text
0  literature   Waves crashed against the shore in roaring ap…
1        tech   A startup introduced a revolutionary battery …
2       quote   Aspire to inspire before we expire, make a ma…
3       quote   Die with memories, not dreams. Live life to i…
4  literature   What we think, we become. Our thoughts shape …
```

2. Split the dataset into training (at least 240examples) and test (at least 60 examples) sets.

[2]:
```
data['Category'] = data['Category'].str.strip()
data['Text'] = data['Text'].str.strip()

data = data.sample(frac=1, random_state=42).reset_index(drop=True)

split_index = int(0.8 * len(data))
train_data = data[:split_index]
test_data = data[split_index:]

(train_data.shape, test_data.shape)
```

[2]:
```
((240, 2), (60, 2))
```

3. Fine-tune a pre-trained language model capable of generating text (e.g., GPT) that you can take, e.g., from the Hugging Face Transformers library.

```
[ ]: !pip install datasets
import tensorflow as tf
from datasets import Dataset
from transformers import AutoTokenizer, TFAutoModelForCausalLM
import pandas as pd
train_dataset = Dataset.from_pandas(train_data)
test_dataset = Dataset.from_pandas(test_data)

tokenizer = AutoTokenizer.from_pretrained("gpt2")
tokenizer.pad_token = tokenizer.eos_token

def tokenize_function(examples):
    return tokenizer(examples['Text'], padding="max_length", truncation=True,␣
 ↪max_length=128)

train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

def map_model_input(input_ids, attention_mask, labels):
    return {"input_ids": input_ids, "attention_mask": attention_mask}, labels

train_tf_dataset = tf.data.Dataset.
 ↪from_tensor_slices((train_dataset['input_ids'],␣
 ↪train_dataset['attention_mask'], train_dataset['input_ids']))
train_tf_dataset = train_tf_dataset.map(map_model_input).shuffle(1000).batch(8)
test_tf_dataset = tf.data.Dataset.
 ↪from_tensor_slices((test_dataset['input_ids'],␣
 ↪test_dataset['attention_mask'], test_dataset['input_ids']))
test_tf_dataset = test_tf_dataset.map(map_model_input).batch(16)

model = TFAutoModelForCausalLM.from_pretrained("gpt2")

optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss,metrics=['accuracy'])

model.fit(train_tf_dataset, epochs=3, validation_data=test_tf_dataset)

eval_loss, eval_acc = model.evaluate(test_tf_dataset)
print(f"Eval Loss: {eval_loss}, Eval Accuracy: {eval_acc}")
```

```
Collecting datasets
  Downloading datasets-2.19.0-py3-none-any.whl (542 kB)
                                     542.0/542.0

kB 7.1 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.13.4)
```

```
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from datasets) (1.25.2)
Requirement already satisfied: pyarrow>=12.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-
packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                           116.3/116.3

kB 10.4 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-
packages (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (4.66.2)
Collecting xxhash (from datasets)
  Downloading
xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                           194.1/194.1

kB 10.4 MB/s eta 0:00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
                           134.8/134.8

kB 11.8 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2024.3.1,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.9.5)
Collecting huggingface-hub>=0.21.2 (from datasets)
  Downloading huggingface_hub-0.22.2-py3-none-any.whl (388 kB)
                           388.9/388.9

kB 10.8 MB/s eta 0:00:00
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (6.0.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.0.5)
```

```
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.21.2->datasets)
(4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.19.0->datasets) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(2024.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.16.0)
Installing collected packages: xxhash, dill, multiprocess, huggingface-hub,
datasets
  Attempting uninstall: huggingface-hub
    Found existing installation: huggingface-hub 0.20.3
    Uninstalling huggingface-hub-0.20.3:
      Successfully uninstalled huggingface-hub-0.20.3
Successfully installed datasets-2.19.0 dill-0.3.8 huggingface-hub-0.22.2
multiprocess-0.70.16 xxhash-3.4.1
```

```
/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), set it as secret in your Google Colab
and restart your session.
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(
```

```
tokenizer_config.json:   0%|          | 0.00/26.0 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/665 [00:00<?, ?B/s]
```

```
vocab.json:   0%|          | 0.00/1.04M [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]

Map:   0%|          | 0/240 [00:00<?, ? examples/s]

Map:   0%|          | 0/60 [00:00<?, ? examples/s]

model.safetensors:   0%|          | 0.00/548M [00:00<?, ?B/s]

All PyTorch model weights were used when initializing TFGPT2LMHeadModel.

All the weights of TFGPT2LMHeadModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFGPT2LMHeadModel for predictions without further training.

Epoch 1/3

WARNING:tensorflow:AutoGraph could not transform <function infer_framework at
0x7c0a9636c8b0> and will run it as-is.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert

WARNING: AutoGraph could not transform <function infer_framework at
0x7c0a9636c8b0> and will run it as-is.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert
30/30 [==============================] - 88s 688ms/step - loss: 1.4431 -
accuracy: 0.7756 - val_loss: 0.1639 - val_accuracy: 0.9689
Epoch 2/3
30/30 [==============================] - 12s 414ms/step - loss: 0.0901 -
accuracy: 0.9807 - val_loss: 0.0417 - val_accuracy: 0.9949
Epoch 3/3
30/30 [==============================] - 13s 423ms/step - loss: 0.0373 -
accuracy: 0.9939 - val_loss: 0.0346 - val_accuracy: 0.9949
4/4 [==============================] - 1s 271ms/step - loss: 0.0346 - accuracy:
0.9949
Eval Loss: 0.03460855782032013, Eval Accuracy: 0.994921863079071
```

1. **Model Architecture and Training Setup**:
   - The model used is TFAutoModelForCausalLM, which is a TensorFlow implementation of the GPT-2 model for language modeling tasks.
   - The optimizer used is Adam with a learning rate of 5e-5.
   - The loss function used is SparseCategoricalCrossentropy, and the model is compiled with the 'accuracy' metric.
2. **Training**:
   - The model is trained for 3 epochs on the training dataset (`train_tf_dataset`).
   - During training, both loss and accuracy metrics are logged for both the training and validation datasets (`test_tf_dataset`).

- As the epochs progress, both training and validation losses decrease, and accuracy increases, indicating that the model is learning and improving its performance.

3. **Evaluation**:
   - After training, the model is evaluated on the test dataset (`test_tf_dataset`).
   - The evaluation metrics show that the model achieves a low evaluation loss of 0.0346 and a high evaluation accuracy of 99.49%.
4. **Analysis**:
   - The model's high evaluation accuracy suggests that it generalizes well to unseen data.
   - The training and validation loss/accuracy curves indicate that the model is not overfitting, as there is no significant gap between the training and validation metrics. Both training and validation loss decrease consistently across epochs, and the validation accuracy reaches a high level comparable to the training accuracy.
5. **Conclusion**:
   - Based on the provided content, there is no evidence of overfitting. The model demonstrates strong performance on both the training and test datasets, with high accuracy and low loss values.
   - The provided code effectively trains and evaluates a GPT-2 model for a language modeling task, showcasing its ability to generate text sequences with high accuracy.

To improve accuracy, consider the following steps:

Fine-Tuning: Fine-tune the GPT-2 model on a downstream task related to your specific domain. This involves training the model on a task-specific dataset to adapt it to your use case.

Hyperparameter Tuning: Experiment with different hyperparameters, such as learning rate, batch size, and model architecture, to find the combination that works best for your data.

Data Augmentation: Increase the diversity of your training data through data augmentation techniques, such as adding noise, paraphrasing, or using different sentence structures.

More Training Data: If possible, obtain more labeled data for training. A larger and more diverse dataset can often lead to better model performance.

Model Architecture: Experiment with different model architectures or try more advanced models that might be better suited for your task.

Regularization: Apply regularization techniques, such as dropout, to prevent overfitting on the training data.

Error Analysis: Analyze the mistakes made by the model on the test set. Identify patterns in misclassifications and consider incorporating this knowledge into the training process.

Good dataset : if possible proper collection of images in data without any distortion,blur, etc.

```
[ ]:
```