TaskA_DistilBERT

November 27, 2024

[1]:

```
import evaluate
import transformers
import numpy as np
import pandas as pd
from datasets import load_dataset, Dataset, DatasetDict
from transformers import (
    AutoTokenizer,
    AutoModelForSequenceClassification,
    TrainingArguments,
    Trainer,
    DistilBertForSequenceClassification,
)
```

```
WARNING:tensorflow:From C:\Users\Admin\miniconda3\Lib\site-
packages\tf_keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

[7]:

```
"""
Download dataset SubtaskA.jsonl from
https://github.com/mbzuai-nlp/M4GT-Bench.
"""
DATA_PATH = "C:/Users/Admin/Desktop/cse847_proj/SubtaskA.jsonl"

# initialize dataset
df = pd.read_json(DATA_PATH, lines=True)
df = df[['text', 'label', 'model']]
dataset = Dataset.from_pandas(df)

# split dataset
a = dataset.train_test_split(test_size=0.20)
b = a['test'].train_test_split(test_size=0.5)
dataset = DatasetDict({
    'train': a['train'],
    'valid': b['train'],
    'test': b['test'],
})
```

```
print(dataset)
```

```
DatasetDict({
    train: Dataset({
        features: ['text', 'label', 'model'],
        num_rows: 122247
    })
    valid: Dataset({
        features: ['text', 'label', 'model'],
        num_rows: 15281
    })
    test: Dataset({
        features: ['text', 'label', 'model'],
        num_rows: 15281
    })
})
```

[14]:

```
print(df.source.value_counts())
print()
print(df.model.value_counts())
```

```
source
wikihow      36556
reddit       33999
arxiv        33998
wikipedia    31365
peerread     16891
Name: count, dtype: int64

model
human      65177
chatGPT    16892
gpt4       14344
davinci    14340
bloomz     14332
dolly      14046
cohere     13678
Name: count, dtype: int64
```

[22]:

```
print(df[df.label == 0].model.value_counts())
print()
print(df[df.label == 1].model.value_counts())
```

```
model
human    65177
Name: count, dtype: int64

model
chatGPT    16892
```

```
gpt4       14344
davinci    14340
bloomz     14332
dolly      14046
cohere     13678
Name: count, dtype: int64
```

[5]:

```python
"""
Initialize tokenizer and model.
"""
model_id = "distilbert-base-uncased"

# init tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)

# init model
model = DistilBertForSequenceClassification.from_pretrained(
    model_id,
    num_labels=2,
)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['classifier.bias', 'classifier.weight', 'pre_classifier.bias',
'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[8]:

```python
"""
Tokenize dataset.
"""
def tokenize(X):
    return tokenizer(
        X["text"],
        padding="max_length",
        truncation=True,
        return_tensors="pt",
    )

# tokenize data
tokenized_datasets = dataset.map(tokenize, batched=True)
print(tokenized_datasets)
```

```
Map:   0%|          | 0/122247 [00:00<?, ? examples/s]

Map:   0%|          | 0/15281 [00:00<?, ? examples/s]

Map:   0%|          | 0/15281 [00:00<?, ? examples/s]

DatasetDict({
    train: Dataset({
        features: ['text', 'label', 'model', 'input_ids', 'attention_mask'],
        num_rows: 122247
    })
    valid: Dataset({
        features: ['text', 'label', 'model', 'input_ids', 'attention_mask'],
        num_rows: 15281
    })
    test: Dataset({
        features: ['text', 'label', 'model', 'input_ids', 'attention_mask'],
        num_rows: 15281
    })
})
```

[12]:

```
"""
Create dataset splits.
"""
seed = 777
n_samples = 10_000
n_test = 1000

train_dataset = tokenized_datasets["train"].shuffle(seed=seed).
 ↪select(range(n_samples))
valid_dataset = tokenized_datasets["valid"].shuffle(seed=seed).
 ↪select(range(n_test))
test_dataset = tokenized_datasets["test"].shuffle(seed=seed).
 ↪select(range(n_test))
```

[13]:

```
"""
Create Trainer.
"""
# define metric
metric = evaluate.load("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

# training args
training_args = TrainingArguments(
    output_dir="C:/Users/Admin/Desktop/cse847_proj/",
    eval_strategy="epoch",
    save_total_limit=3,
)

# init trainer
```

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=valid_dataset,
    compute_metrics=compute_metrics,
)
```

[14]:

```
"""
Train model.
"""
trainer.train()
```

```
<IPython.core.display.HTML object>
```

[14]:

```
TrainOutput(global_step=3750, training_loss=0.13993426310221355,
metrics={'train_runtime': 33095.9456, 'train_samples_per_second': 0.906,
'train_steps_per_second': 0.113, 'total_flos': 3974021959680000.0, 'train_loss':
0.13993426310221355, 'epoch': 3.0})
```

[15]:

```
"""
Evaluate trained model.
"""
trainer.evaluate(test_dataset)
```

```
<IPython.core.display.HTML object>
```

[15]:

```
{'eval_loss': 0.3407599627971649,
 'eval_accuracy': 0.941,
 'eval_runtime': 305.5325,
 'eval_samples_per_second': 3.273,
 'eval_steps_per_second': 0.409,
 'epoch': 3.0}
```

[16]:

```
"""
Summarize model.
"""
print(model)
```

```
DistilBertForSequenceClassification(
  (distilbert): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): DistilBertSdpaAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
        )
      )
    )
  )
  (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[ ]: