Supervised_stance_detection

November 28, 2023

1 Stance Detection: Hyperparameter sweeps with Weights and Biases

This tutorial demonstrates how to train a transformer with hyperparameter sweep. Our task is to classify support for Trump in a data set of tweets, and we will use a RoBERTa model that has been domain adapted for classification of political tweets.

Requirements:

1. A basic understanding of Python.
2. Access to a GPU (free services like Google Colab work well).
3. An account with Weights and Biases.

For additional resources, consult the documentation. Simple Transformers

Weights and Biases

[1]:
```
# import the necessary libraries
import pandas as pd
from simpletransformers.classification import ClassificationModel
from simpletransformers.classification import ClassificationArgs
import torch
import wandb
```

First we need to check that we can properly communicate with the GPU. If the code prints "Cuda available", everything is working as expected. If you are using Google Colab and this returns CPU, then click on "Runtime" at the top of the page, select "Change Runtime", and then select "GPU" under "Hardware Accelerator."

[2]:
```
print("Cuda available" if torch.cuda.is_available() is True else "CPU")
print("PyTorch version: ", torch.__version__)
```

```
Cuda available
PyTorch version:  1.10.2
```

For this example we will use a data set consists of tweets about President Trump that are manually labeled 1: approve, 0: not approve.

[3]:
```
train_df = pd.read_csv('https://raw.githubusercontent.com/XXXXX/
↪stance_detection_tutorials/main/data/train.csv')
```

```
test_df = pd.read_csv('https://raw.githubusercontent.com/XXXXX/
↪stance_detection_tutorials/main/data/test.csv')
```

[4]:
```
test_df
```

[4]:
```
                                                   text  labels
0     @realDonaldTrump  I like Mexicans who come to ...       1
1     RT @AhmedtheBanker: Let's not forget @realDona...       0
2     @realDonaldTrump did not apply to immigrants o...       0
3     Been slacking on my @realDonaldTrump retweets...       1
4     And how many  #latinos enemies you gained in 1...       0
...                                                 ...     ...
1130  yes @USER preach !!!! america is listing!!!!!#...       1
1131  tonight i will go to bed thinking about the ab...       0
1132  @USER \n\ncome ny harlem, long island...reassu...       1
1133  @USER trump battered that scary boogie man bid...       1
1134  so did chris wallace or @USER win the debate??...       0

[1135 rows x 2 columns]
```

This will login to Weights and Biases so that we can send results to our account and track our experiment.

[5]:
```
wandb.login()
```

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server
locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here:
https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press
ctrl+c to quit:

  ........................................

wandb: Appending key for api.wandb.ai to your netrc file:
C:\Users\XXXXX/.netrc
```

[5]:
```
True
```

This chunk of code configues our sweep parameters. We can define how we want to search the hyperparameter space (bayes, grid, or random), as well as the range of parameters we want to search. If using a bayesian sweep, we can choose the performance metric we want to optimize for. In this case, I'm using a bayesian sweep to maximize MCC.

Generally, the most important parameters to search are the number of training epochs (how many times the training data passes through the model) and the learning rate (the size of the changes implement when the models adjusts its weights and biases during training).

[6]:
```
sweep_config = {
    "method": "bayes",
```

```
    "metric": {"name": "mcc", "goal": "maximize"},
    "parameters": {
        "num_train_epochs": {"min": 0, "max": 10},
        "learning_rate": {"min": 0.0, "max": 5e-04 },
    },
#    "early_terminate": {"type": "hyperband", "min_iter":6,}, # early terminate␣
 ↪can be used to stop runs of the model that have turned degenerate.
}
```

Using our configuration, we can now initialize the sweep. This will create a project on your Weights and Biases account and pass your sweep configuration to that project.

[7]:
```
sweep_id = wandb.sweep(sweep_config, project="stance_sweep")
```

```
Create sweep with ID: ktzjgwry
Sweep URL: https://wandb.ai/XXXXX/stance_sweep/sweeps/ktzjgwry
```

Next we define the specific parameters we want to use in the actual model.

[8]:
```
model_args = ClassificationArgs()

model_args.evaluate_during_training = True # Whether or not the model should␣
 ↪periodically stop to evaluate performance during training
model_args.evaluate_during_training_silent = True
model_args.evaluate_during_training_steps = 10 # How frequently the model should␣
 ↪stop
model_args.manual_seed = 1 # Random seed
model_args.max_seq_length = 512 # Max sequence length is the maximum number of␣
 ↪tokens the model can accept. Anything beyond the max length is truncated.
model_args.train_batch_size = 16 # The batch size is the number of examples that␣
 ↪are passed through the model at a time. Larger batch sizes train faster, but␣
 ↪require more VRAM on the GPU.
model_args.eval_batch_size = 16
model_args.train_custom_parameters_only = False
model_args.wandb_project = "stance_sweep"
# Adjust these args if you want to save local copies of the model. This is␣
 ↪generally not recommended as models are large and many models can quickly fill␣
 ↪all of your disk space.
model_args.no_cache = True
model_args.no_save = True
model_args.overwrite_output_dir = True
model_args.reprocess_input_data = True
```

The final step before starting the sweep is to define our training function. Within the function we choose which model we are using, pass the model and sweep configuration, and tell the model which data to use.

[9]:
```
# training function
def train():
    # Initialize a new wandb run
    wandb.init(resume = True)

    # Create a TransformerModel
    model = ClassificationModel(
        model_type = "bertweet", # Model type refers to the base version of the␣
↪model. In this case we are using BERTweet, a RoBERTa model trained on Twitter␣
↪data.

        # Model name refers to the specific version of the model we want to use.
        # In this case, we are using a version of BERTweet that has been domain␣
↪adapted for political tweets found here: https://huggingface.co/kornosk/
↪polibertweet-political-twitter-roberta-mlm.
        # You can use different models by pointing it to a different model on␣
↪the huggingface model repository.
        model_name = "kornosk/polibertweet-political-twitter-roberta-mlm",
        weight = [1,3], # We're going to use weights since our data set is␣
↪unbalanced, there are about three times as many not support tweets as support.␣
↪This will weight missclassification of support tweets more heavily when␣
↪calculating the model's loss or error.
        use_cuda=True, # Use the GPU
        args=model_args, # pass model arguments we defined above to the function
        sweep_config=wandb.config, # Pass sweep configuration we defined above␣
↪to the function
    )

    # Train the model
    model.train_model(
        train_df,
        eval_df=test_df,
        verbose = False,
        accuracy=lambda truth, predictions: accuracy_score(
            truth, [round(p) for p in predictions]
        ),
    )

    # Sync with W&B
    wandb.join()
```

Now we start training the model by telling the wandb agent which sweep to run, which training function to use, and how many iterations of the model it should train. Results should update in realtime on your W&B account!

[ ]:
```
wandb.agent(sweep_id, train, count = 30)
```

```
wandb: Agent Starting Run: 55m91mm7 with config:
```

```
wandb:     learning_rate: 0.0001880319350438253
wandb:     num_train_epochs: 0
wandb: Currently logged in as: XXXXX. Use `wandb
login --relogin` to force relogin
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
Downloading:   0%|          | 0.00/697 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/515M [00:00<?, ?B/s]
```