

## Homework 9

In this homework, you will train a sentiment classifier on the [SST-2](#) dataset using the pre-trained BERT model. For simplicity, use the [Hugging Face Transformers library](#). There are two parts in this week's homework:

- Part 1: Fine-tuning an Encoder-only Transformer

1. Fine-tune [TinyBERT](#) on SST-2 and evaluate the results. You can find a tutorial for loading BERT and fine-tuning [here](#). In the tutorial, you will need to change the dataset from "yelp\_review\_full" to "sst2" and the model from "bert-base-uncased" to "huawei-noah/TinyBERT\_General\_L4L312D". You will also need to modify the code since SST-2 is a two-class classification dataset (unlike the Yelp Reviews dataset, which is a five-class classification dataset).
2. Choose a different pre-trained BERT-style model from the [Hugging Face Model Hub](#) and fine-tune it. There are tons of options - part of the homework is navigating the hub to find different models! We recommend picking a model that is smaller than BERT-Base (as TinyBERT is) just to make things computationally cheaper. Is the final validation accuracy higher or lower with this other model?

```
# pip install datasets transformers numpy evaluate
```

```
def tokenize(examples):
    return tokenizer(examples["sentence"], padding="max_length", truncation=True, max_length=5)
```

```
from datasets import load_dataset
from transformers import AutoTokenizer
```

```
model_name = "huawei-noah/TinyBERT_General_4L_312D"
```

```
dataset = load_dataset("sst2")
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
dataset = dataset.map(tokenize, batched=True)
```

```

/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: 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
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(

```

```
small_train = dataset["train"].shuffle(seed=42).select(range(5000))
small_eval = dataset["train"].shuffle(seed=42).select(range(5001, 10001))
# small_eval = dataset["test"].shuffle(seed=42).select(range(1000))
```

```
print(set(small_train["label"]))
print(set(small_eval["label"]))
```

$$\Rightarrow \begin{matrix} \{0, 1\} \\ \{0, 1\} \end{matrix}$$

```
from transformers import AutoModelForSequenceClassification
```

```
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)
```

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at huawei-noah/TinyBERT\_General\_4L\_312D and You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
import numpy as np
import evaluate

metric = evaluate.load("accuracy")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    # convert the logits to their predicted class
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

```
from transformers import TrainingArguments
```

```
training_args = TrainingArguments(
    output_dir="tinyBERT_sst",
    eval_strategy="epoch",
    push_to_hub=False,
)
```

```
from transformers import Trainer
```

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

```
trainer.train()
```

🔗 **wandb:** WARNING The `run\_name` is currently set to the same value as `TrainingArguments.output\_dir`. If this was not intended, please specify a different name for the run.

**wandb:** Using wandb-core as the SDK backend. Please refer to <https://wandb.me/wandb-core> for more information.

**wandb:** Currently logged in as: **skylar-t-furey** ([skylar-t-furey-the-university-of-north-carolina-at-chape](https://wandb.ai/skylar-t-furey-the-university-of-north-carolina-at-chape)) to <https://api.wandb.ai>. Use `wandb login` to view your profile page.

Tracking run with wandb version 0.19.8

Run data is saved locally in `/content/wandb/run-20250327_001120-izucm4yx`

Syncing run **tinyBERT\_sst** to [Weights & Biases \(docs\)](https://wandb.ai/skylar-t-furey-the-university-of-north-carolina-at-chape/huggingface)

View project at <https://wandb.ai/skylar-t-furey-the-university-of-north-carolina-at-chape/huggingface>

View run at <https://wandb.ai/skylar-t-furey-the-university-of-north-carolina-at-chape/huggingface/runs/izucm4yx>

[1875/1875 00:45, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.564200	0.503455	0.738600
2	0.502200	0.518958	0.756800
3	0.449600	0.537998	0.757600

TrainOutput(global\_step=1875, training\_loss=0.49008367513020834, metrics={'train\_runtime': 48.2259, 'train\_samples\_per\_second': 38.462, 'val\_runtime': 1.1111, 'val\_samples\_per\_second': 16.667, 'epoch': 3})

```
model_name2 = "FacebookAI/roberta-base"
```

```
model2 = AutoModelForSequenceClassification.from_pretrained(model_name2, num_labels=2)
```

```
trainer2 = Trainer(
    model=model2,
    args=training_args,
    train_dataset=small_train,
    eval_dataset=small_eval,
    compute_metrics=compute_metrics,
)
```

```
trainer2.train()
```

🔗 Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at FacebookAI/roberta-base and are newly initialized from the random normal distribution. You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

[1875/1875 03:18, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.696400	0.688591	0.551800
2	0.694300	0.687822	0.551800
3	0.689800	0.687810	0.551800

TrainOutput(global\_step=1875, training\_loss=0.692601171875, metrics={'train\_runtime': 198.3881, 'train\_samples\_per\_second': 75.609, 'val\_runtime': 1.1111, 'val\_samples\_per\_second': 16.667, 'epoch': 3})

## Comparison

The RoBERTa base model created by Facebook AI is much worse at classifying the SST2 dataset. Around 50% accuracy is especially bad considering there are only two labels.

## ✓ Part 2: Encoder-only Transformers as Feature Extractors

Instead of fine-tuning the full model on a target dataset, it's also possible to simply use the output representations from a BERT-style model as input to a linear classifier and *only* train the classifier (leaving the rest of the pre-trained parameters fixed). You can do this easily using the [sentence-transformers](#) library. Using `sentence-transformers` gives you back a fixed-length representation of a given text sequence. To achieve this, you need to

1. Pick a pre-trained sentence Transformer.
2. Load the SST-2 dataset and feed the text from each example into the model.
3. Train a linear classifier on the representations.
4. Evaluate performance on the validation set.

For the second step, you can learn more about how to use Hugging Face datasets [here](#). For the third and fourth step, you can do this directly in PyTorch, or you can just collect the learned representations and use them as feature vectors to train a linear classifier in any other library (e.g. [scikit-learn](#)).

After you complete the above steps, report whether the accuracy on the validation set is higher or lower using a fixed sentence Transformer.

```
from sentence_transformers import SentenceTransformer

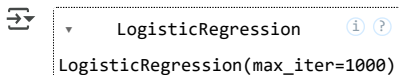
model = SentenceTransformer("all-MiniLM-L6-v2")

X_train = model.encode(small_train["sentence"])
y_train = small_train["label"]

X_val = model.encode(small_eval["sentence"])
y_val = small_eval["label"]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


# Train logistic regression
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train, y_train)
```



LogisticRegression  
LogisticRegression(max\_iter=1000)

```
y_pred = clf.predict(X_val)

# Compute accuracy
accuracy = accuracy_score(y_val, y_pred)
print(f"Validation Accuracy: {accuracy:.4f}")
```



Validation Accuracy: 0.8364

## Comparison

The sentence transformer was the best of all three methods at classifying the SST2 dataset with a validation accuracy of ~84%.