50.055 Machine Learning Operations Hands On Session Notes

Authored by: Daniel Dahlmeier, January 2023.

Session 2: Train BERT classifier for sentiment classification in Jupyter Notebook

Objective

In this session, you will delve into fine-tuning a BERT model for classification. BERT is one of the most influential models of modern NLP. We will apply it to the sentiment classification problem from the last session. This will give us a direct comparison of "traditional" NLP and "modern NLP with foundation models. We will do the fine-tuning in an interactive Jupyter notebook environment.

Understanding BERT

BERT is a pre-trained transformer encoder model which can be finetuned for downstream tasks, For an introduction to BERT, check out this Huggingfance blog:
https://huggingface.co/blog/bert-101

1. Start Jupyter notebook

Start a Jupyter notebook on the SUTD cluster.

2. Checkout Github repository

Check out the Github repo by opening a terminal and run

```
~$ git clone https://github.com/ddahlmeier/sutd-mlops-course-code.git
```

Open the notebook **02_finetune_bert_notebook.ipynb** in the folder sutd-mlops-course-code.

3. Development environment, login to Weights & Biases

Run the first cells to install all required dependencies, import the necessary packages and log in to your wandb account. All this should feel familiar from the last hands-on session.

4. Data Preparation

Load the Rotten Tomatoes dataset again.

~~5. Feature engineering~~

Unlike the linear model, the BERT deep learning model learns the feature representations itself. Thus, we have no feature engineering step.

6. Tokenize data

A tokenizer separates text into individual tokens, usually a word or subset of a word. It is important to use the correct tokenizer. Luckily Huggingface makes it easy to load the correct tokenizer for a model with the AutoTokenizer.from_pretrained() method.
We will be using the DistilBERT model and therefore need to load its associated tokenizer.

```
model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)
```

```
tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

7. Subsample data

Training deep learning models is a lot slower than training linear models. To reduce the time for this hands-on exercise, we select a random subsample of the training, evaluation, and test data.

8. Load model

Load the DistilBERT model from Huggingface model hub. DistilBERT is a smaller version of BERT which requires less memory and is faster to train.

You can learn more about DistilBERT here: https://huggingface.co/papers/1910.01108

```
from transformers import AutoModelForSequenceClassification

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

9. Train model

We use the Hugginface Trainer class to start the model training with the required training arguments and evaluate . We log training metrics to wandb.

10. Test model

Make predictions on the training, validation and test set and compute the accuracy.

Open wandb and inspect the logged metrics and plots of the experiment.

11. Experimenting with hyperparameters

The Huggingface Trainer class has a lot of hyperparameters. If you have time, explore different parameters and their effect, for example number of epochs, batch size, or learning rate

If you have time, re-run the experiment with the full data set and with the original BERT model instead of DistillBERT.

12. Cleanup

Please clean up all resources.

- Shut down the kernel and logout of the SUTD Education Cluster

Conclusion

This session provided hands-on experience with finetuning a BERT classifier for sentiment classification. Now you can finetune state-of-the-art NLP models.

Additional Resources

- BERT research paper
- DistilBERT research paper