model_06_bert_cross_encoder_classification_grid_search_2

May 8, 2023

1 Model 06 Bert Cross Entropy Classification for Label Prediction

Prediction of claim labels based on the matched evidence.

1.1 Setup

1.1.1 Working Directory

[ ]:
```
# Change the working directory to project root
from pathlib import Path
import os
ROOT_DIR = Path.cwd()
while not ROOT_DIR.joinpath("src").exists():
    ROOT_DIR = ROOT_DIR.parent
os.chdir(ROOT_DIR)
```

1.1.2 Dependencies

[ ]:
```
# Imports and dependencies
import torch
from torch.utils.data import DataLoader
from torch.nn import CrossEntropyLoss
from torch.optim import AdamW
from torch.optim.lr_scheduler import LinearLR
from torcheval.metrics import MulticlassAccuracy, MulticlassF1Score

from src.logger import SimpleLogger
from src.model_05 import BertCrossEncoderClassifier
from src.data import LabelClassificationDataset
from src.torch_utils import get_torch_device
import json
from dataclasses import dataclass
from typing import List, Union, Tuple
from tqdm import tqdm
import random
import numpy as np
from datetime import datetime
from math import exp
```

```
from sklearn.model_selection import ParameterGrid

TORCH_DEVICE = get_torch_device()
```

```
/opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

```
Torch device is 'mps'
```

1.1.3 File paths

[ ]:
```
MODEL_PATH = ROOT_DIR.joinpath("./result/models/*")
DATA_PATH = ROOT_DIR.joinpath("./data/*")
LOG_PATH = ROOT_DIR.joinpath("./result/logs/*")
SHORTLIST_PATH = ROOT_DIR.joinpath("./result/pipeline/shortlisting_v2/*")

run_time = datetime.now().strftime('%Y_%m_%d_%H_%M')
```

1.2 Training Loop

[ ]:
```
def training_loop(
    model,
    claims_paths:List[Path],
    save_path:Path=None,
    warmup:float=0.1,
    lr:float=0.00005, # 5e-5
    weight_decay:float=0.01,
    normalize_text:bool=True,
    max_length:int=128,
    dropout:float=None,
    n_epochs:int=5,
    batch_size:int=64,
):
    # Generate training dataset
    train_data = LabelClassificationDataset(
        claims_paths=claims_paths,
        training=True,
    )
    train_dataloader = DataLoader(
        dataset=train_data,
        shuffle=True,
        batch_size=batch_size
    )
```

```
    # Generate evaluation dataset
    dev_data = LabelClassificationDataset(
        claims_paths=[Path("./data/dev-claims.json")],
        training=True,
    )
    dev_dataloader = DataLoader(
        dataset=dev_data,
        shuffle=False,
        batch_size=batch_size
    )

    # Loss function
    loss_fn = CrossEntropyLoss()

    # Optimizer
    optimizer = AdamW(
        params=model.parameters(),
        lr=lr,
        weight_decay=weight_decay
    )

    # Scheduler
    scheduler = LinearLR(
        optimizer=optimizer,
        total_iters=warmup * len(train_dataloader),
        verbose=False
    )

    # Metrics
    accuracy_fn = MulticlassAccuracy()
    f1_fn = MulticlassF1Score()

    # Training epochs ------------------------------------------------------

    best_epoch_loss = 999
    best_epoch_f1 = -1
    best_epoch_acc = -1
    best_epoch = 0
    for epoch in range(n_epochs):

        print(f"Epoch: {epoch + 1} of {n_epochs}\n")

        # Run training ---------------------------------------------------
        model.train()

        train_batches = tqdm(train_dataloader, desc="train batches")
        running_losses = []
```

```
        for batch in train_batches:
            claim_texts, evidence_texts, labels, claim_ids, evidence_ids = batch
            texts = list(zip(claim_texts, evidence_texts))

            # Reset optimizer
            optimizer.zero_grad()

            # Forward + loss
            output, logits, seq = model(
                texts=texts,
                normalize_text=normalize_text,
                max_length=max_length,
                dropout=dropout
            )
            loss = loss_fn(logits, labels)

            # Backward + optimizer
            loss.backward()
            optimizer.step()

            # Update running loss
            batch_loss = loss.item() * len(batch)
            running_losses.append(batch_loss)

            train_batches.postfix = f"loss: {batch_loss:.3f}"

            # Update scheduler
            scheduler.step()

            continue

        # Epoch loss
        epoch_loss = np.average(running_losses)
        print(f"Average epoch loss: {epoch_loss:.3f}")

        # Run evaluation ----------------------------------------------------
        model.eval()

        dev_batches = tqdm(dev_dataloader, desc="dev batches")
        dev_acc = []
        dev_f1 = []
        for batch in dev_batches:
            claim_texts, evidence_texts, labels, claim_ids, evidence_ids = batch
            texts = list(zip(claim_texts, evidence_texts))

            # Forward
            output, logits, seq = model(
```

```
                texts=texts,
                normalize_text=normalize_text,
                max_length=max_length,
                dropout=dropout
            )

            # Prediction
            predicted = torch.argmax(output, dim=1)

            # Metrics
            accuracy_fn.update(predicted.cpu(), labels.cpu())
            f1_fn.update(predicted.cpu(), labels.cpu())

            acc = accuracy_fn.compute()
            f1 = f1_fn.compute()

            dev_acc.append(acc)
            dev_f1.append(f1)

            dev_batches.postfix = f" acc: {acc:.3f}, f1: {f1:.3f}"

            continue

        # Consider metrics
        epoch_acc = np.average(dev_acc)
        print(f"Average epoch accuracy: {epoch_acc:.3f}")

        epoch_f1 = np.average(dev_f1)
        print(f"Average epoch f1: {epoch_f1:.3f}")

        if epoch_acc > best_epoch_acc:
            best_epoch_acc = epoch_acc

        if epoch_f1 > best_epoch_f1:
            best_epoch_f1 = epoch_f1
            best_epoch = epoch + 1

        # Save model ---------------------------------------------------------

        # Save the model with the best f1 score
        if save_path and epoch_f1 >= best_epoch_f1:
            torch.save(model, save_path)
            print(f"Saved model to: {save_path}")

    print("Done!")
    return best_epoch_acc, best_epoch_f1, best_epoch
```

1.3 Load model

Use a blank pre-trained

[ ]:
```
model = BertCrossEncoderClassifier(
    pretrained_name="bert-base-uncased",
    n_classes=3,
    device=TORCH_DEVICE
)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.weight',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).
```

Or load one previously trained

[ ]:
```
# MODEL_SAVE_PATH = MODEL_PATH.with_name("")
# with open(MODEL_PATH.with_name(MODEL_SAVE_PATH), mode="rb") as f:
#     model = torch.load(f, map_location=TORCH_DEVICE)
```

1.4 Training and evaluation loop

[ ]:
```
# training_loop(
#     model=model,
#     claims_paths=[
#         DATA_PATH.with_name("train-claims.json")
#     ],
#     save_path=MODEL_PATH.
 ↪with_name(f"model_06_bert_cross_encoder_label_{run_time}.pth"),
#     warmup=0.1,
#     lr=0.00005, # 5e-5
#     weight_decay=0.01,
#     normalize_text=True,
#     max_length=512,
#     dropout=None,
#     n_epochs=1,
#     batch_size=24,
# )
```

1.5 Tune hyperparameters

[ ]:
```
hyperparams = ParameterGrid(param_grid={
    "claims_paths": [[
        DATA_PATH.with_name("train-claims.json")
    ]],
    "warmup": [0.1],
    "lr": [0.000005],
    "weight_decay": [0.02],
    "normalize_text": [True],
    "max_length": [512],
    "dropout": [0.1, 0.2],
    "n_epochs": [3],
    "batch_size": [8, 32],
    "freeze_bert": [False]
})
```

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
with SimpleLogger("model_06_cross_encoder_retrieval") as logger:
    logger.set_stream_handler()
    logger.set_file_handler(
        log_path=LOG_PATH,
        filename="model_06_hyperparam_tuning.txt"
    )
    best_f1 = -1
    best_params = {}
    for hyperparam in hyperparams:
        model = BertCrossEncoderClassifier(
            pretrained_name="bert-base-uncased",
            n_classes=3,
            device=TORCH_DEVICE
        )

        model_param = hyperparam.copy()

        # Freeze bert parameters if desired
        if "freeze_bert" in model_param.keys():
            if hyperparam["freeze_bert"] is True:
                for param in model.bert.parameters():
                    param.requires_grad = False
            del model_param["freeze_bert"]

        logger.info("\n== RUN")
        logger.info(hyperparam)
```

```
        accuracy, f1, epoch = training_loop(model=model, **model_param)

        logger.info(f"run_best_epoch: {epoch}, run_best_acc: {accuracy},␣
↪run_best_f1: {f1}")

        if f1 > best_f1:
            best_f1 = f1
            best_params = hyperparam

        logger.info(f"\n== CURRENT BEST F1: {best_f1}")
        logger.info(best_params)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.weight',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

2023-05-08 13:47:33 model_06_cross_encoder_retrieval:INFO

== RUN

2023-05-08 13:47:33 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.1, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}

Torch device is 'mps'

claims: 100%|      | 1228/1228 [00:00<00:00, 475411.23it/s]

generated dataset n=3730
Torch device is 'mps'

claims: 100%|      | 154/154 [00:00<00:00, 544622.95it/s]

generated dataset n=433
Epoch: 1 of 3


train batches: 100%|      | 467/467 [03:47<00:00,  2.05it/s, loss: 1.060]

Average epoch loss: 4.479
```

```
dev batches: 100%|     | 55/55 [00:07<00:00,  7.35it/s,  acc: 0.564, f1:
0.564]

Average epoch accuracy: 0.601
Average epoch f1: 0.601
Epoch: 2 of 3


train batches: 100%|     | 467/467 [03:47<00:00,  2.06it/s, loss: 1.061]

Average epoch loss: 3.305

dev batches: 100%|     | 55/55 [00:07<00:00,  7.50it/s,  acc: 0.558, f1:
0.558]

Average epoch accuracy: 0.569
Average epoch f1: 0.569
Epoch: 3 of 3


train batches: 100%|     | 467/467 [03:46<00:00,  2.06it/s, loss: 0.201]

Average epoch loss: 2.120

dev batches: 100%|     | 55/55 [00:07<00:00,  7.57it/s,  acc: 0.570, f1:
0.570]

Average epoch accuracy: 0.570
Average epoch f1: 0.570
Done!
2023-05-08 13:59:18 model_06_cross_encoder_retrieval:INFO
run_best_epoch: 1, run_best_acc: 0.601498007774353, run_best_f1:
0.601498007774353

2023-05-08 13:59:18 model_06_cross_encoder_retrieval:INFO

== CURRENT BEST F1: 0.601498007774353

2023-05-08 13:59:18 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.1, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}


Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.weight',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
```

```
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

2023-05-08 13:59:20 model_06_cross_encoder_retrieval:INFO

== RUN

2023-05-08 13:59:20 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.2, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}

Torch device is 'mps'

claims: 100%|     | 1228/1228 [00:00<00:00, 441392.18it/s]

generated dataset n=3730
Torch device is 'mps'

claims: 100%|     | 154/154 [00:00<00:00, 618700.02it/s]

generated dataset n=433
Epoch: 1 of 3


train batches: 100%|     | 467/467 [03:47<00:00,  2.05it/s, loss: 4.158]

Average epoch loss: 4.537

dev batches: 100%|     | 55/55 [00:07<00:00,  7.55it/s,  acc: 0.561, f1:
0.561]

Average epoch accuracy: 0.616
Average epoch f1: 0.616
Epoch: 2 of 3


train batches: 100%|     | 467/467 [03:48<00:00,  2.05it/s, loss: 3.084]

Average epoch loss: 3.305

dev batches: 100%|     | 55/55 [00:07<00:00,  7.56it/s,  acc: 0.581, f1:
0.581]

Average epoch accuracy: 0.585
Average epoch f1: 0.585
Epoch: 3 of 3


train batches: 100%|     | 467/467 [03:48<00:00,  2.05it/s, loss: 4.307]

Average epoch loss: 2.327
```

```
dev batches: 100%|      | 55/55 [00:07<00:00,  7.54it/s,  acc: 0.587, f1:
0.587]

Average epoch accuracy: 0.591
Average epoch f1: 0.591
Done!
2023-05-08 14:11:07 model_06_cross_encoder_retrieval:INFO
run_best_epoch: 1, run_best_acc: 0.6160757541656494, run_best_f1:
0.6160757541656494

2023-05-08 14:11:07 model_06_cross_encoder_retrieval:INFO

== CURRENT BEST F1: 0.6160757541656494

2023-05-08 14:11:07 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.2, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}


Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.weight',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

2023-05-08 14:11:08 model_06_cross_encoder_retrieval:INFO

== RUN

2023-05-08 14:11:08 model_06_cross_encoder_retrieval:INFO
{'batch_size': 32, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.1, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}

Torch device is 'mps'

claims: 100%|      | 1228/1228 [00:00<00:00, 631046.96it/s]

generated dataset n=3730
Torch device is 'mps'
```

```
claims: 100%|      | 154/154 [00:00<00:00, 621079.63it/s]

generated dataset n=433
Epoch: 1 of 3


train batches: 100%|      | 117/117 [03:22<00:00,  1.73s/it, loss: 4.321]

Average epoch loss: 4.887

dev batches: 100%|      | 14/14 [00:07<00:00,  1.97it/s,  acc: 0.527, f1:
0.527]

Average epoch accuracy: 0.580
Average epoch f1: 0.580
Epoch: 2 of 3


train batches: 100%|      | 117/117 [03:22<00:00,  1.73s/it, loss: 2.214]

Average epoch loss: 4.020

dev batches: 100%|      | 14/14 [00:06<00:00,  2.00it/s,  acc: 0.527, f1:
0.527]

Average epoch accuracy: 0.546
Average epoch f1: 0.546
Epoch: 3 of 3


train batches: 100%|      | 117/117 [03:21<00:00,  1.72s/it, loss: 3.609]

Average epoch loss: 3.360

dev batches: 100%|      | 14/14 [00:06<00:00,  2.00it/s,  acc: 0.530, f1:
0.530]

Average epoch accuracy: 0.539
Average epoch f1: 0.539
Done!
2023-05-08 14:21:37 model_06_cross_encoder_retrieval:INFO
run_best_epoch: 1, run_best_acc: 0.5803744196891785, run_best_f1:
0.5803744196891785

2023-05-08 14:21:37 model_06_cross_encoder_retrieval:INFO

== CURRENT BEST F1: 0.6160757541656494

2023-05-08 14:21:37 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.2, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when
initializing BertModel: ['cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.weight',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of
a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

2023-05-08 14:21:39 model_06_cross_encoder_retrieval:INFO

== RUN

2023-05-08 14:21:39 model_06_cross_encoder_retrieval:INFO
{'batch_size': 32, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.2, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}

Torch device is 'mps'

claims: 100%|     | 1228/1228 [00:00<00:00, 601635.94it/s]

generated dataset n=3730
Torch device is 'mps'

claims: 100%|     | 154/154 [00:00<00:00, 628941.40it/s]

generated dataset n=433
Epoch: 1 of 3


train batches: 100%|     | 117/117 [03:23<00:00,  1.74s/it, loss: 4.077]

Average epoch loss: 4.923

dev batches: 100%|     | 14/14 [00:06<00:00,  2.00it/s,  acc: 0.439, f1:
0.439]

Average epoch accuracy: 0.450
Average epoch f1: 0.450
Epoch: 2 of 3


train batches: 100%|     | 117/117 [03:21<00:00,  1.72s/it, loss: 2.892]

Average epoch loss: 4.342
```

```
dev batches: 100%|     | 14/14 [00:06<00:00,  2.01it/s,  acc: 0.498, f1:
0.498]

Average epoch accuracy: 0.491
Average epoch f1: 0.491
Epoch: 3 of 3


train batches: 100%|     | 117/117 [03:21<00:00,  1.72s/it, loss: 2.412]

Average epoch loss: 3.807

dev batches: 100%|     | 14/14 [00:06<00:00,  2.00it/s,  acc: 0.517, f1:
0.517]

Average epoch accuracy: 0.518
Average epoch f1: 0.518
Done!
2023-05-08 14:32:07 model_06_cross_encoder_retrieval:INFO
run_best_epoch: 3, run_best_acc: 0.5175042748451233, run_best_f1:
0.5175042748451233

2023-05-08 14:32:07 model_06_cross_encoder_retrieval:INFO

== CURRENT BEST F1: 0.6160757541656494

2023-05-08 14:32:07 model_06_cross_encoder_retrieval:INFO
{'batch_size': 8, 'claims_paths':
[PosixPath('/Users/johnsonzhou/git/comp90042-project/data/train-claims.json')],
'dropout': 0.2, 'freeze_bert': False, 'lr': 5e-06, 'max_length': 512,
'n_epochs': 3, 'normalize_text': True, 'warmup': 0.1, 'weight_decay': 0.02}
```