model_01_er_siamese_bert_2023_04_27_22_45

April 28, 2023

1 Model 01

Evidence retrieval using a Siamese BERT classification model.

Ref: - STS continue training guide

1.1 Setup

1.1.1 Working Directory

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

1.1.2 File paths

[ ]:
```
MODEL_PATH = ROOT_DIR.joinpath("./result/models/*")
```

1.1.3 Dependencies

[ ]:
```
# Imports and dependencies
import spacy
import torch
from torch import nn
from torch.utils.data import DataLoader
from sentence_transformers import SentenceTransformer, LoggingHandler
from sentence_transformers.losses import SoftmaxLoss
from sentence_transformers.evaluation import BinaryClassificationEvaluator
from src.torch_utils import get_torch_device
from src.spacy_utils import process_sentence
from src.model_01 import ClaimEvidenceDataset
from datetime import datetime
import logging
import math
```

```
torch_device = get_torch_device()
```

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

```
/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
```

1.1.4 Names

[ ]:
```
run_time = datetime.now().strftime('%Y_%m_%d_%H_%M')
model_save_path = MODEL_PATH.with_name(f"model_01_{run_time}")
eval_name = "model_01_dev"
```

1.1.5 Logging

[ ]:
```
logging.basicConfig(format='%(asctime)s - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S',
    level=logging.INFO,
    handlers=[LoggingHandler()]
)
```

1.2 Dataset

[ ]:
```
train_data = ClaimEvidenceDataset(
    claims_json="./data/train-claims.json",
    evidence_json="./data/evidence.json",
    negative_sample_strategy="related_random",
    negative_sample_size=500,
    preprocess_func=None
)
dev_data = ClaimEvidenceDataset(
    claims_json="./data/dev-claims.json",
    evidence_json="./data/evidence.json",
    negative_sample_strategy="related_random",
    negative_sample_size=500,
    preprocess_func=None
)
```

```
Generate claim-evidence pair with related_random strategy n=500
```

```
claims: 100%|      | 1228/1228 [01:20<00:00, 15.20it/s]
```

```
Generate claim-evidence pair with related_random strategy n=500
```

```
claims: 100%|      | 154/154 [00:08<00:00, 17.74it/s]
```

[ ]:
```
print(len(train_data))
print(len(dev_data))
```

```
1232122
148302
```

[ ]:
```
# for sample in train_data:
#     if sample.texts[0] == "Not only is there no scientific evidence that CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support more plant and animal life.":
#         print(sample)
```

1.3 Select model components

[ ]:
```
nlp = spacy.load("en_core_web_trf")
nlp
```

[ ]:
```
<spacy.lang.en.English at 0x28a379550>
```

[ ]:
```
model = SentenceTransformer(
    "sentence-transformers/msmarco-bert-base-dot-v5",
    device=torch_device
)
model
```

```
2023-04-27 22:47:16 - Load pretrained SentenceTransformer: sentence-transformers/msmarco-bert-base-dot-v5
```

[ ]:
```
SentenceTransformer(
  (0): Transformer({'max_seq_length': 512, 'do_lower_case': False}) with Transformer model: BertModel
  (1): Pooling({'word_embedding_dimension': 768, 'pooling_mode_cls_token': False, 'pooling_mode_mean_tokens': True, 'pooling_mode_max_tokens': False, 'pooling_mode_mean_sqrt_len_tokens': False})
)
```

[ ]:
```
train_loss = SoftmaxLoss(
    model=model,
    sentence_embedding_dimension=model.get_sentence_embedding_dimension(),
    num_labels=2,
    concatenation_sent_rep=True,
    concatenation_sent_difference=True,
    concatenation_sent_multiplication=False,
    loss_fct=nn.CrossEntropyLoss()
)
train_loss
```

```
2023-04-27 22:47:27 - Softmax loss: #Vectors concatenated: 3
```

[ ]:
```
SoftmaxLoss(
  (model): SentenceTransformer(
    (0): Transformer({'max_seq_length': 512, 'do_lower_case': False}) with
Transformer model: BertModel
    (1): Pooling({'word_embedding_dimension': 768, 'pooling_mode_cls_token':
False, 'pooling_mode_mean_tokens': True, 'pooling_mode_max_tokens': False,
'pooling_mode_mean_sqrt_len_tokens': False})
  )
  (classifier): Linear(in_features=2304, out_features=2, bias=True)
  (loss_fct): CrossEntropyLoss()
)
```

[ ]:
```
train_eval = BinaryClassificationEvaluator.from_input_examples(
    examples=dev_data,
    name=eval_name,
    write_csv=True,
    show_progress_bar=True
)
train_eval
```

[ ]:
```
<sentence_transformers.evaluation.BinaryClassificationEvaluator.BinaryClassifica
tionEvaluator at 0x2c49bbaf0>
```

1.4 Training

[ ]:
```
train_batch_size = 64
num_epochs = 5
```

[ ]:
```
train_dataloader = DataLoader(
    dataset=train_data,
    shuffle=True,
    batch_size=train_batch_size
)
dev_dataloader = DataLoader(
    dataset=dev_data,
    shuffle=True,
    batch_size=train_batch_size
)
```

[ ]:
```
#10% of train data for warm-up
warmup_steps = math.ceil(len(train_dataloader) * num_epochs * 0.1)
```

[ ]:
```
# Train the model
model.fit(
    train_objectives=[(train_dataloader, train_loss)],
```

```
    epochs=num_epochs,
    evaluator=train_eval,
    evaluation_steps=1000,
    warmup_steps=warmup_steps,
    optimizer_class=torch.optim.AdamW,
    optimizer_params={"lr": 0.00002},
    weight_decay=0.01,
    output_path=str(model_save_path),
    save_best_model=True,
    show_progress_bar=True
)
```

```
Epoch:   0%|          | 0/5 [00:00<?, ?it/s]/opt/homebrew/Caskroom/miniconda/bas
e/envs/comp90042_project/lib/python3.8/site-
packages/torch/autograd/__init__.py:200: UserWarning: The operator
'aten::sgn.out' is not currently supported on the MPS backend and will fall back
to run on the CPU. This may have performance implications. (Triggered internally
at /Users/runner/miniforge3/conda-bld/pytorch-
recipe_1680607560203/work/aten/src/ATen/mps/MPSFallback.mm:11.)
  Variable._execution_engine.run_backward(  # Calls into the C++ engine to run
the backward pass

Epoch:   0%|          | 0/5 [38:34<?, ?it/s]

2023-04-27 23:27:15 - Binary Accuracy Evaluation of the model on model_01_dev
dataset in epoch 0 after 1000 steps:
```

```
Batches: 100%|     | 2341/2341 [01:59<00:00, 19.55it/s]
```

```
Epoch:   0%|          | 0/5 [40:37<?, ?it/s]
2023-04-27 23:29:18 - Accuracy with Cosine-Similarity:          99.67
(Threshold: 0.9851)
2023-04-27 23:29:18 - F1 with Cosine-Similarity:                13.26
(Threshold: 0.9738)
2023-04-27 23:29:18 - Precision with Cosine-Similarity:         21.56
2023-04-27 23:29:18 - Recall with Cosine-Similarity:            9.57
2023-04-27 23:29:18 - Average Precision with Cosine-Similarity: 5.60
```

```
Epoch:   0%|          | 0/5 [40:37<?, ?it/s]
2023-04-27 23:29:19 - Accuracy with Manhattan-Distance:          99.67
(Threshold: 59.5647)
2023-04-27 23:29:19 - F1 with Manhattan-Distance:                12.97
(Threshold: 78.7452)
2023-04-27 23:29:19 - Precision with Manhattan-Distance:         20.09
2023-04-27 23:29:19 - Recall with Manhattan-Distance:            9.57
2023-04-27 23:29:19 - Average Precision with Manhattan-Distance: 5.44
```

```
Epoch:   0%|          | 0/5 [40:38<?, ?it/s]
```

```
2023-04-27 23:29:19 - Accuracy with Euclidean-Distance:           99.67
(Threshold: 2.6911)
2023-04-27 23:29:19 - F1 with Euclidean-Distance:                 12.98
(Threshold: 3.5669)
2023-04-27 23:29:19 - Precision with Euclidean-Distance:          20.17
2023-04-27 23:29:19 - Recall with Euclidean-Distance:             9.57
2023-04-27 23:29:19 - Average Precision with Euclidean-Distance:  5.44
```

```
Epoch:   0%|          | 0/5 [40:38<?, ?it/s]
2023-04-27 23:29:19 - Accuracy with Dot-Product:           99.67
(Threshold: 252.7120)
2023-04-27 23:29:19 - F1 with Dot-Product:                 5.44 (Threshold:
239.2893)
2023-04-27 23:29:19 - Precision with Dot-Product:          3.84
2023-04-27 23:29:19 - Recall with Dot-Product:             9.37
2023-04-27 23:29:19 - Average Precision with Dot-Product:  2.02

2023-04-27 23:29:19 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_22_45
```

```
Epoch:   0%|          | 0/5 [1:11:12<?, ?it/s]
2023-04-27 23:59:53 - Binary Accuracy Evaluation of the model on model_01_dev
dataset in epoch 0 after 2000 steps:
```

```
Batches: 100%|      | 2341/2341 [01:34<00:00, 24.87it/s]
```

```
Epoch:   0%|          | 0/5 [1:12:50<?, ?it/s]

2023-04-28 00:01:31 - Accuracy with Cosine-Similarity:           99.67
(Threshold: 0.9813)
2023-04-28 00:01:31 - F1 with Cosine-Similarity:                 14.54
(Threshold: 0.9632)
2023-04-28 00:01:31 - Precision with Cosine-Similarity:          14.23
2023-04-28 00:01:31 - Recall with Cosine-Similarity:             14.87
2023-04-28 00:01:31 - Average Precision with Cosine-Similarity:  6.95
```

```
Epoch:   0%|          | 0/5 [1:12:50<?, ?it/s]

2023-04-28 00:01:31 - Accuracy with Manhattan-Distance:           99.67
(Threshold: 67.3073)
2023-04-28 00:01:31 - F1 with Manhattan-Distance:                 14.43
(Threshold: 94.4607)
2023-04-28 00:01:31 - Precision with Manhattan-Distance:          14.20
2023-04-28 00:01:31 - Recall with Manhattan-Distance:             14.66
2023-04-28 00:01:32 - Average Precision with Manhattan-Distance:  6.87
```

```
Epoch:   0%|          | 0/5 [1:12:51<?, ?it/s]

2023-04-28 00:01:32 - Accuracy with Euclidean-Distance:           99.67
(Threshold: 3.0587)
2023-04-28 00:01:32 - F1 with Euclidean-Distance:                 14.71
(Threshold: 4.2692)
2023-04-28 00:01:32 - Precision with Euclidean-Distance:          14.75
2023-04-28 00:01:32 - Recall with Euclidean-Distance:             14.66
2023-04-28 00:01:32 - Average Precision with Euclidean-Distance:  6.82
```

```
Epoch:   0%|          | 0/5 [1:12:51<?, ?it/s]
```

```
2023-04-28 00:01:32 - Accuracy with Dot-Product:           99.67
(Threshold: 255.0250)
2023-04-28 00:01:32 - F1 with Dot-Product:                 10.99
(Threshold: 243.0355)
2023-04-28 00:01:32 - Precision with Dot-Product:          8.01
2023-04-28 00:01:32 - Recall with Dot-Product:             17.52
2023-04-28 00:01:32 - Average Precision with Dot-Product:  5.27

2023-04-28 00:01:32 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_22_45


Epoch:   0%|          | 0/5 [1:35:03<?, ?it/s]

2023-04-28 00:23:44 - Binary Accuracy Evaluation of the model on model_01_dev
dataset in epoch 0 after 3000 steps:
```

```
Batches: 100%|     | 2341/2341 [01:34<00:00, 24.82it/s]
```

```
Epoch:   0%|          | 0/5 [1:36:40<?, ?it/s]
2023-04-28 00:25:21 - Accuracy with Cosine-Similarity:          99.67
(Threshold: 0.9890)
2023-04-28 00:25:21 - F1 with Cosine-Similarity:                15.45
(Threshold: 0.9745)
2023-04-28 00:25:21 - Precision with Cosine-Similarity:         17.16
2023-04-28 00:25:21 - Recall with Cosine-Similarity:            14.05
2023-04-28 00:25:21 - Average Precision with Cosine-Similarity:  7.03
```

```
Epoch:   0%|          | 0/5 [1:36:41<?, ?it/s]

2023-04-28 00:25:22 - Accuracy with Manhattan-Distance:           99.67
(Threshold: 53.9586)
2023-04-28 00:25:22 - F1 with Manhattan-Distance:                 15.16
(Threshold: 81.3441)
2023-04-28 00:25:22 - Precision with Manhattan-Distance:          17.05
2023-04-28 00:25:22 - Recall with Manhattan-Distance:             13.65
2023-04-28 00:25:22 - Average Precision with Manhattan-Distance:  7.06
```

```
Epoch:   0%|          | 0/5 [1:36:41<?, ?it/s]

2023-04-28 00:25:22 - Accuracy with Euclidean-Distance:           99.67
(Threshold: 2.4397)
2023-04-28 00:25:22 - F1 with Euclidean-Distance:                 15.45
(Threshold: 3.7775)
2023-04-28 00:25:22 - Precision with Euclidean-Distance:          15.42
2023-04-28 00:25:22 - Recall with Euclidean-Distance:             15.48
2023-04-28 00:25:22 - Average Precision with Euclidean-Distance:  7.12
```

```
Epoch:   0%|          | 0/5 [1:36:41<?, ?it/s]

2023-04-28 00:25:23 - Accuracy with Dot-Product:           99.67
(Threshold: 269.8538)
2023-04-28 00:25:23 - F1 with Dot-Product:                 9.46 (Threshold:
261.7487)
2023-04-28 00:25:23 - Precision with Dot-Product:          7.29
2023-04-28 00:25:23 - Recall with Dot-Product:             13.44
2023-04-28 00:25:23 - Average Precision with Dot-Product:  3.98

2023-04-28 00:25:23 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_22_45

Iteration:  19%|         | 3737/19252 [1:48:11<7:29:10,  1.74s/it]
Epoch:   0%|          | 0/5 [1:48:11<?, ?it/s]
```

```
RuntimeError                              Traceback (most recent call last)
Cell In[16], line 2
      1 # Train the model
----> 2 model.fit(
      3     train_objectives=[(train_dataloader, train_loss)],
      4     epochs=num_epochs,
      5     evaluator=train_eval,
      6     evaluation_steps=1000,
      7     warmup_steps=warmup_steps,
      8     optimizer_class=torch.optim.AdamW,
      9     optimizer_params={"lr": 0.00002},
     10     weight_decay=0.01,
     11     output_path=str(model_save_path),
     12     save_best_model=True,
     13     show_progress_bar=True
     14 )

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/sentence_transformers/SentenceTransformer.py:721, in␣
 ↪SentenceTransformer.fit(self, train_objectives, evaluator, epochs,␣
 ↪steps_per_epoch, scheduler, warmup_steps, optimizer_class, optimizer_params,␣
 ↪weight_decay, evaluation_steps, output_path, save_best_model, max_grad_norm,␣
 ↪use_amp, callback, show_progress_bar, checkpoint_path, checkpoint_save_steps,
 ↪checkpoint_save_total_limit)
    719     skip_scheduler = scaler.get_scale() != scale_before_step
    720 else:
--> 721     loss_value = loss_model(features, labels)
    722     loss_value.backward()
    723     torch.nn.utils.clip_grad_norm_(loss_model.parameters(),␣
 ↪max_grad_norm)

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/sentence_transformers/losses/SoftmaxLoss.py:62, in SoftmaxLoss.
 ↪forward(self, sentence_features, labels)
     61 def forward(self, sentence_features: Iterable[Dict[str, Tensor]], label
 ↪ Tensor):
```

```
---> 62     reps = [self.model(sentence_feature)['sentence_embedding'] for␣
 ↪sentence_feature in sentence_features]
     63     rep_a, rep_b = reps
     65     vectors_concat = []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/sentence_transformers/losses/SoftmaxLoss.py:62, in <listcomp>(.)
     61 def forward(self, sentence_features: Iterable[Dict[str, Tensor]], label
 ↪ Tensor):
---> 62     reps = [self.model(sentence_feature)['sentence_embedding'] for␣
 ↪sentence_feature in sentence_features]
     63     rep_a, rep_b = reps
     65     vectors_concat = []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/container.py:217, in Sequential.forward(self,␣
 ↪input)
    215 def forward(self, input):
    216     for module in self:
--> 217         input = module(input)
    218     return input

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []
```

```
File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/sentence_transformers/models/Transformer.py:66, in Transformer.
 ↪forward(self, features)
     63 if 'token_type_ids' in features:
     64     trans_features['token_type_ids'] = features['token_type_ids']
---> 66 output_states = self.auto_model(**trans_features, return_dict=False)
     67 output_tokens = output_states[0]
     69 features.update({'token_embeddings': output_tokens, 'attention_mask':␣
 ↪features['attention_mask']})

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/transformers/models/bert/modeling_bert.py:1014, in BertModel.
 ↪forward(self, input_ids, attention_mask, token_type_ids, position_ids,␣
 ↪head_mask, inputs_embeds, encoder_hidden_states, encoder_attention_mask,␣
 ↪past_key_values, use_cache, output_attentions, output_hidden_states,␣
 ↪return_dict)
   1005 head_mask = self.get_head_mask(head_mask, self.config.num_hidden_layers
   1007 embedding_output = self.embeddings(
   1008     input_ids=input_ids,
   1009     position_ids=position_ids,
   (…)
   1012     past_key_values_length=past_key_values_length,
   1013 )
-> 1014 encoder_outputs = self.encoder(
   1015     embedding_output,
   1016     attention_mask=extended_attention_mask,
   1017     head_mask=head_mask,
   1018     encoder_hidden_states=encoder_hidden_states,
   1019     encoder_attention_mask=encoder_extended_attention_mask,
   1020     past_key_values=past_key_values,
   1021     use_cache=use_cache,
   1022     output_attentions=output_attentions,
   1023     output_hidden_states=output_hidden_states,
   1024     return_dict=return_dict,
   1025 )
   1026 sequence_output = encoder_outputs[0]
```

```
   1027 pooled_output = self.pooler(sequence_output) if self.pooler is not None
 ↪else None

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/transformers/models/bert/modeling_bert.py:603, in BertEncoder.
 ↪forward(self, hidden_states, attention_mask, head_mask, encoder_hidden_states␣
 ↪encoder_attention_mask, past_key_values, use_cache, output_attentions,␣
 ↪output_hidden_states, return_dict)
    594     layer_outputs = torch.utils.checkpoint.checkpoint(
    595         create_custom_forward(layer_module),
    596         hidden_states,
   (…)
    600         encoder_attention_mask,
    601     )
    602 else:
--> 603     layer_outputs = layer_module(
    604         hidden_states,
    605         attention_mask,
    606         layer_head_mask,
    607         encoder_hidden_states,
    608         encoder_attention_mask,
    609         past_key_value,
    610         output_attentions,
    611     )
    613 hidden_states = layer_outputs[0]
    614 if use_cache:

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
```

```
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/transformers/models/bert/modeling_bert.py:489, in BertLayer.
 ↪forward(self, hidden_states, attention_mask, head_mask, encoder_hidden_states
 ↪encoder_attention_mask, past_key_value, output_attentions)
    477 def forward(
    478     self,
    479     hidden_states: torch.Tensor,
   (…)
    486 ) -> Tuple[torch.Tensor]:
    487     # decoder uni-directional self-attention cached key/values tuple is
 ↪at positions 1,2
    488     self_attn_past_key_value = past_key_value[:2] if past_key_value is
 ↪not None else None
--> 489     self_attention_outputs = self.attention(
    490         hidden_states,
    491         attention_mask,
    492         head_mask,
    493         output_attentions=output_attentions,
    494         past_key_value=self_attn_past_key_value,
    495     )
    496     attention_output = self_attention_outputs[0]
    498     # if decoder, the last output is tuple of self-attn cache

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/transformers/models/bert/modeling_bert.py:419, in BertAttention
 ↪forward(self, hidden_states, attention_mask, head_mask, encoder_hidden_states
 ↪encoder_attention_mask, past_key_value, output_attentions)
    409 def forward(
    410     self,
    411     hidden_states: torch.Tensor,
   (…)
    417     output_attentions: Optional[bool] = False,
```

```
    418 ) -> Tuple[torch.Tensor]:
--> 419     self_outputs = self.self(
    420         hidden_states,
    421         attention_mask,
    422         head_mask,
    423         encoder_hidden_states,
    424         encoder_attention_mask,
    425         past_key_value,
    426         output_attentions,
    427     )
    428     attention_output = self.output(self_outputs[0], hidden_states)
    429     outputs = (attention_output,) + self_outputs[1:]  # add attentions␣
 ↪if we output them

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self,␣
 ↪*args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File /opt/homebrew/Caskroom/miniconda/base/envs/comp90042_project/lib/python3.8
 ↪site-packages/transformers/models/bert/modeling_bert.py:359, in␣
 ↪BertSelfAttention.forward(self, hidden_states, attention_mask, head_mask,␣
 ↪encoder_hidden_states, encoder_attention_mask, past_key_value,␣
 ↪output_attentions)
    355     attention_probs = attention_probs * head_mask
    357 context_layer = torch.matmul(attention_probs, value_layer)
--> 359 context_layer = context_layer.permute(0, 2, 1, 3).contiguous()
    360 new_context_layer_shape = context_layer.size()[:-2] + (self.
 ↪all_head_size,)
    361 context_layer = context_layer.view(new_context_layer_shape)

RuntimeError: MPS backend out of memory (MPS allocated: 40.27 GB, other␣
 ↪allocations: 82.52 GB, max allowed: 122.40 GB). Tried to allocate 89.06 MB on
 ↪private pool. Use PYTORCH_MPS_HIGH_WATERMARK_RATIO=0.0 to disable upper limit
 ↪for memory allocations (may cause system failure).
```