model_01_er_siamese_bert_2023_04_27

April 27, 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

[ ]:
```
DATA_PATH = ROOT_DIR.joinpath("./result/train_data/*")
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

[ ]:
```
# Path to the claim evidence pair json data file
train_data_file = DATA_PATH.with_name("train_claim_evidence_pair_rns.json")
dev_data_file = DATA_PATH.with_name("dev_claim_evidence_pair_rns.json")
```

[ ]:
```
train_data = ClaimEvidenceDataset(
    claims_json="./data/train-claims.json",
    evidence_json="./data/evidence.json",
    negative_sample_strategy="related_random",
    negative_sample_size=0,
    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=0,
    preprocess_func=None
)
```

```
Generate claim-evidence pair with related_random strategy n=0
```

```
claims: 100%|      | 1228/1228 [01:19<00:00, 15.43it/s]

Generate claim-evidence pair with related_random strategy n=0

claims: 100%|      | 154/154 [00:08<00:00, 18.52it/s]
```

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

```
12366
1473
```

[ ]:
```
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)
```

```
<InputExample> label: 1, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; At very high concentrations (100 times atmospheric
concentration, or greater), carbon dioxide can be toxic to animal life, so
raising the concentration to 10,000 ppm (1%) or higher for several hours will
eliminate pests such as whiteflies and spider mites in a greenhouse.
<InputExample> label: 1, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; Plants can grow as much as 50 percent faster in
concentrations of 1,000 ppm CO 2 when compared with ambient conditions, though
this assumes no change in climate and no limitation on other nutrients.
<InputExample> label: 1, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; Higher carbon dioxide concentrations will
favourably affect plant growth and demand for water.
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; Since water vapor is a greenhouse gas, the increase
in water vapor content makes the atmosphere warm further; this warming causes
the atmosphere to hold still more water vapor (a positive feedback), and so on
until other processes stop the feedback loop.
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; As the Earth's climate warms, we are seeing many
changes: stronger, more destructive hurricanes; heavier rainfall; more
disastrous flooding; more areas of the world experiencing severe drought; and
more heat waves."
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; Between 1989 and 2002 the Global Climate Coalition,
a group of mainly United States businesses, used aggressive lobbying and public
```

```
relations tactics to oppose action to reduce greenhouse gas emissions and fight
the Kyoto Protocol.
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; The Guia Race of Macau is an international touring
car race, and currently a round of the TCR International Series.
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; Also on the property is a barn built about 1900.
<InputExample> label: 0, texts: Not only is there no scientific evidence that
CO2 is a pollutant, higher CO2 concentrations actually help ecosystems support
more plant and animal life.; The rank listed is the recipient 's rank at the
time the Knight 's Cross was awarded.
```

1.3 Select model components

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

[ ]:
```
<spacy.lang.en.English at 0x10640c580>
```

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

```
2023-04-27 14:59:17 - 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 14:59:17 - 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 0x2ca2c5910>
```

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/base/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
Iteration: 100%|     | 194/194 [06:31<00:00,  2.02s/it]
Epoch:   0%|          | 0/5 [06:31<?, ?it/s]

2023-04-27 15:05:49 - Binary Accuracy Evaluation of the model on model_01_dev dataset after epoch 0:

Batches: 100%|     | 35/35 [00:08<00:00,  4.28it/s]
Epoch:   0%|          | 0/5 [06:39<?, ?it/s]

2023-04-27 15:05:57 - Accuracy with Cosine-Similarity:          87.98
(Threshold: 0.7984)
2023-04-27 15:05:57 - F1 with Cosine-Similarity:                81.77
(Threshold: 0.7973)
2023-04-27 15:05:57 - Precision with Cosine-Similarity:         82.71
2023-04-27 15:05:57 - Recall with Cosine-Similarity:            80.86
2023-04-27 15:05:57 - Average Precision with Cosine-Similarity:  87.95

2023-04-27 15:05:57 - Accuracy with Manhattan-Distance:          88.32
(Threshold: 233.5319)
2023-04-27 15:05:57 - F1 with Manhattan-Distance:                82.48
(Threshold: 233.5319)
2023-04-27 15:05:57 - Precision with Manhattan-Distance:         82.48
2023-04-27 15:05:57 - Recall with Manhattan-Distance:            82.48
2023-04-27 15:05:57 - Average Precision with Manhattan-Distance:  88.22
```

```
2023-04-27 15:05:57 - Accuracy with Euclidean-Distance:           88.19
(Threshold: 10.7523)
2023-04-27 15:05:57 - F1 with Euclidean-Distance:                 82.32
(Threshold: 10.7523)
2023-04-27 15:05:57 - Precision with Euclidean-Distance:          82.15
2023-04-27 15:05:57 - Recall with Euclidean-Distance:             82.48
2023-04-27 15:05:57 - Average Precision with Euclidean-Distance:  88.11

2023-04-27 15:05:57 - Accuracy with Dot-Product:           86.22
(Threshold: 221.0562)
2023-04-27 15:05:57 - F1 with Dot-Product:                 80.89
(Threshold: 199.2893)
2023-04-27 15:05:57 - Precision with Dot-Product:          74.28
2023-04-27 15:05:57 - Recall with Dot-Product:             88.80
2023-04-27 15:05:57 - Average Precision with Dot-Product:  84.85

2023-04-27 15:05:57 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_14_57

Iteration: 100%|     | 194/194 [08:56<00:00,  2.77s/it]
Epoch:  20%|         | 1/5 [15:36<26:41, 400.32s/it]

2023-04-27 15:14:54 - Binary Accuracy Evaluation of the model on model_01_dev
dataset after epoch 1:

Batches: 100%|     | 35/35 [00:01<00:00, 20.69it/s]
Epoch:  20%|         | 1/5 [15:38<26:41, 400.32s/it]

2023-04-27 15:14:56 - Accuracy with Cosine-Similarity:           90.70
(Threshold: 0.6919)
2023-04-27 15:14:56 - F1 with Cosine-Similarity:                 86.14
(Threshold: 0.6649)
2023-04-27 15:14:56 - Precision with Cosine-Similarity:          83.82
2023-04-27 15:14:56 - Recall with Cosine-Similarity:             88.59
2023-04-27 15:14:56 - Average Precision with Cosine-Similarity:  92.78

2023-04-27 15:14:56 - Accuracy with Manhattan-Distance:           90.90
(Threshold: 292.0865)
2023-04-27 15:14:56 - F1 with Manhattan-Distance:                 86.72
(Threshold: 310.7213)
2023-04-27 15:14:56 - Precision with Manhattan-Distance:          83.30
2023-04-27 15:14:56 - Recall with Manhattan-Distance:             90.43
2023-04-27 15:14:56 - Average Precision with Manhattan-Distance:  92.88

2023-04-27 15:14:56 - Accuracy with Euclidean-Distance:           90.77
(Threshold: 13.6339)
2023-04-27 15:14:56 - F1 with Euclidean-Distance:                 86.42
(Threshold: 14.3041)
2023-04-27 15:14:56 - Precision with Euclidean-Distance:          84.17
2023-04-27 15:14:56 - Recall with Euclidean-Distance:             88.80
```

```
2023-04-27 15:14:56 - Average Precision with Euclidean-Distance:  92.87

2023-04-27 15:14:56 - Accuracy with Dot-Product:           90.36
(Threshold: 205.5502)
2023-04-27 15:14:56 - F1 with Dot-Product:                 85.77
(Threshold: 205.5502)
2023-04-27 15:14:56 - Precision with Dot-Product:          84.42
2023-04-27 15:14:56 - Recall with Dot-Product:             87.17
2023-04-27 15:14:56 - Average Precision with Dot-Product:  92.47

2023-04-27 15:14:56 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_14_57

Iteration: 100%|     | 194/194 [06:03<00:00,  1.87s/it]
Epoch:  40%|        | 2/5 [21:42<24:05, 481.81s/it]

2023-04-27 15:21:00 - Binary Accuracy Evaluation of the model on model_01_dev
dataset after epoch 2:

Batches: 100%|     | 35/35 [00:01<00:00, 19.94it/s]
Epoch:  60%|       | 3/5 [21:44<14:17, 428.70s/it]

2023-04-27 15:21:02 - Accuracy with Cosine-Similarity:           89.88
(Threshold: 0.6594)
2023-04-27 15:21:02 - F1 with Cosine-Similarity:                 84.95
(Threshold: 0.6281)
2023-04-27 15:21:02 - Precision with Cosine-Similarity:          83.20
2023-04-27 15:21:02 - Recall with Cosine-Similarity:             86.76
2023-04-27 15:21:02 - Average Precision with Cosine-Similarity:  92.05

2023-04-27 15:21:02 - Accuracy with Manhattan-Distance:           90.16
(Threshold: 299.4382)
2023-04-27 15:21:02 - F1 with Manhattan-Distance:                 85.46
(Threshold: 320.8383)
2023-04-27 15:21:02 - Precision with Manhattan-Distance:          83.08
2023-04-27 15:21:02 - Recall with Manhattan-Distance:             87.98
2023-04-27 15:21:02 - Average Precision with Manhattan-Distance:  92.33

2023-04-27 15:21:02 - Accuracy with Euclidean-Distance:           89.95
(Threshold: 14.4851)
2023-04-27 15:21:02 - F1 with Euclidean-Distance:                 85.02
(Threshold: 16.8200)
2023-04-27 15:21:02 - Precision with Euclidean-Distance:          78.25
2023-04-27 15:21:02 - Recall with Euclidean-Distance:             93.08
2023-04-27 15:21:02 - Average Precision with Euclidean-Distance:  92.14

2023-04-27 15:21:02 - Accuracy with Dot-Product:           89.75
(Threshold: 195.2626)
2023-04-27 15:21:02 - F1 with Dot-Product:                 84.95
(Threshold: 195.2626)
```

```
2023-04-27 15:21:02 - Precision with Dot-Product:          83.20
2023-04-27 15:21:02 - Recall with Dot-Product:             86.76
2023-04-27 15:21:02 - Average Precision with Dot-Product:  91.73


Iteration: 100%|     | 194/194 [05:52<00:00,  1.82s/it]
Epoch:  60%|       | 3/5 [27:36<14:17, 428.70s/it]

2023-04-27 15:26:54 - Binary Accuracy Evaluation of the model on model_01_dev
dataset after epoch 3:

Batches: 100%|     | 35/35 [00:01<00:00, 20.82it/s]
Epoch:  60%|       | 3/5 [27:38<14:17, 428.70s/it]

2023-04-27 15:26:56 - Accuracy with Cosine-Similarity:           90.56
(Threshold: 0.6651)
2023-04-27 15:26:56 - F1 with Cosine-Similarity:                 86.37
(Threshold: 0.5710)
2023-04-27 15:26:56 - Precision with Cosine-Similarity:          81.67
2023-04-27 15:26:56 - Recall with Cosine-Similarity:             91.65
2023-04-27 15:26:56 - Average Precision with Cosine-Similarity:  92.78

2023-04-27 15:26:56 - Accuracy with Manhattan-Distance:           90.90
(Threshold: 314.1307)
2023-04-27 15:26:56 - F1 with Manhattan-Distance:                 86.90
(Threshold: 340.7839)
2023-04-27 15:26:56 - Precision with Manhattan-Distance:          82.45
2023-04-27 15:26:56 - Recall with Manhattan-Distance:             91.85
2023-04-27 15:26:56 - Average Precision with Manhattan-Distance:  92.95

2023-04-27 15:26:56 - Accuracy with Euclidean-Distance:           90.63
(Threshold: 15.5071)
2023-04-27 15:26:56 - F1 with Euclidean-Distance:                 86.54
(Threshold: 16.3056)
2023-04-27 15:26:56 - Precision with Euclidean-Distance:          81.97
2023-04-27 15:26:56 - Recall with Euclidean-Distance:             91.65
2023-04-27 15:26:56 - Average Precision with Euclidean-Distance:  92.84

2023-04-27 15:26:56 - Accuracy with Dot-Product:           90.43
(Threshold: 206.5302)
2023-04-27 15:26:56 - F1 with Dot-Product:                 86.29
(Threshold: 182.2049)
2023-04-27 15:26:56 - Precision with Dot-Product:          82.02
2023-04-27 15:26:56 - Recall with Dot-Product:             91.04
2023-04-27 15:26:56 - Average Precision with Dot-Product:  92.73

2023-04-27 15:26:56 - Save model to
/Users/johnsonzhou/git/comp90042-project/result/models/model_01_2023_04_27_14_57

Iteration: 100%|     | 194/194 [04:32<00:00,  1.40s/it]
```

```
Epoch:  80%|      | 4/5 [32:11<06:39, 399.44s/it]

2023-04-27 15:31:29 - Binary Accuracy Evaluation of the model on model_01_dev
dataset after epoch 4:

Batches: 100%|     | 35/35 [00:01<00:00, 19.71it/s]
Epoch: 100%|     | 5/5 [32:13<00:00, 386.65s/it]

2023-04-27 15:31:31 - Accuracy with Cosine-Similarity:           90.70
(Threshold: 0.6251)
2023-04-27 15:31:31 - F1 with Cosine-Similarity:                 86.37
(Threshold: 0.6251)
2023-04-27 15:31:31 - Precision with Cosine-Similarity:          84.44
2023-04-27 15:31:31 - Recall with Cosine-Similarity:             88.39
2023-04-27 15:31:31 - Average Precision with Cosine-Similarity:  92.75

2023-04-27 15:31:31 - Accuracy with Manhattan-Distance:           91.04
(Threshold: 316.1617)
2023-04-27 15:31:31 - F1 with Manhattan-Distance:                 86.77
(Threshold: 316.1617)
2023-04-27 15:31:31 - Precision with Manhattan-Distance:          85.40
2023-04-27 15:31:31 - Recall with Manhattan-Distance:             88.19
2023-04-27 15:31:31 - Average Precision with Manhattan-Distance:  92.90

2023-04-27 15:31:31 - Accuracy with Euclidean-Distance:           90.77
(Threshold: 15.2621)
2023-04-27 15:31:31 - F1 with Euclidean-Distance:                 86.50
(Threshold: 15.8968)
2023-04-27 15:31:31 - Precision with Euclidean-Distance:          83.24
2023-04-27 15:31:31 - Recall with Euclidean-Distance:             90.02
2023-04-27 15:31:31 - Average Precision with Euclidean-Distance:  92.80

2023-04-27 15:31:31 - Accuracy with Dot-Product:           90.56
(Threshold: 195.8921)
2023-04-27 15:31:31 - F1 with Dot-Product:                 86.19
(Threshold: 185.7503)
2023-04-27 15:31:31 - Precision with Dot-Product:          82.50
2023-04-27 15:31:31 - Recall with Dot-Product:             90.22
2023-04-27 15:31:31 - Average Precision with Dot-Product:  92.71
```