TEMPLATE

October 24, 2025

[1]:
```
%reload_ext autoreload
%autoreload 2
```

[2]:
```
from kret_studies import *
from kret_studies.notebook import *
from kret_studies.complex import *

logger = get_notebook_logger()
```

```
INFO:datasets:JAX version 0.7.2 available.
```

```
Loaded environment variables from /Users/Akseldkw/coding/Columbia/UML-
Project/.env.
```

```
INFO:arviz.preview:arviz_base not installed
INFO:arviz.preview:arviz_stats not installed
INFO:arviz.preview:arviz_plots not installed
```

```
/Users/Akseldkw/coding/kretsinger/data/nb_log.log
```

[3]:
```
from uml_project import *

HF_DIR, HF_REGISTRY, DEVICE_TORCH
```

[3]:
```
(PosixPath('/Users/Akseldkw/coding/Columbia/UML-Project/data/huggingface'),
 PosixPath('/Users/Akseldkw/coding/Columbia/UML-
Project/data/huggingface/REGISTRY.json'),
 device(type='mps'))
```

[4]:
```
IMDB_DIR = HF_DIR / "imdb"
```

[5]:
```
df_imdb_train = pd.read_parquet(IMDB_DIR / "train.parquet")
df_imdb_test = pd.read_parquet(IMDB_DIR / "test.parquet")
```

[6]:
```
df = df_imdb_train
test_df = df_imdb_test
```

[ ]:
```
stsb_dict = load_dataset("glue", "stsb")
```

[23]:
```
df_stsb_train: pd.DataFrame = stsb_dict["train"].to_pandas()  # type: ignore
df_stsb_val: pd.DataFrame = stsb_dict["validation"].to_pandas()  # type: ignore
df_stsb_test: pd.DataFrame = stsb_dict["test"].to_pandas()  # type: ignore
```

[31]:
```
# df_stsb_train.sort_values("label", ascending=False)
# df_stsb_val.sort_values("label", ascending=False)
# df_stsb_test.sort_values("label", ascending=False)
```

[10]:
```
datasets = list(huggingface_hub.list_datasets(dataset_name="stsb"))
```

[11]:
```
word_emb = models.Transformer("bert-base-uncased")
pooling = models.Pooling(word_emb.get_word_embedding_dimension(),␣
 ↪pooling_mode_mean_tokens=True)
dense = models.Dense(
    in_features=word_emb.get_word_embedding_dimension(), out_features=128,␣
 ↪activation_function=torch.nn.Tanh()
)

model = SentenceTransformer(modules=[word_emb, pooling, dense])
```

[12]:
```
BASE_MODEL = "sentence-transformers/all-MiniLM-L6-v2"  # ABOBA: small, fast
TARGET_DIM = 64  # ABOBA desired embedding dimensionality (experiment with 32,
# 64, 128...)
BATCH_SIZE = 64
POOLER_LR = 2e-4
FINETUNE_LR = 2e-5
```

[13]:
```
EPOCHS_POOLER = 2  # step A epochs (pooler only)
EPOCHS_FINETUNE = 2  # step B epochs (unfreeze and train)
```

[ ]:
```
def build_model(base_model_name: str, target_dim: int):
    """
    Build a SentenceTransformer where we append a Dense projection after pooling
    to obtain exactly `target_dim` output dimensions.
    """
    # Transformer (encoder)
    word_embedding_model = models.Transformer(base_model_name,␣
 ↪max_seq_length=128)
    # Mean pooling (or use cls pooling if you prefer)
    pooling_model = models.Pooling(
        word_embedding_model.get_word_embedding_dimension(),
        pooling_mode_mean_tokens=True,  # sentence embedding = mean of word
        # embeddings in sentence, that's rule of thumb for sentence similarity␣
 ↪but if
        # we want to do classification prob cls is better
        pooling_mode_cls_token=False,  # instead of cls or max use mean here;
        # ABOBA: can vary and see changes
```

```
        pooling_mode_max_tokens=False,
    )
    # The pooler (projector)
    dense = models.Dense(
        in_features=pooling_model.get_sentence_embedding_dimension(),
        out_features=target_dim,
        activation_function=nn.Tanh(),
    )
    model = SentenceTransformer(modules=[word_embedding_model, pooling_model,␣
↪dense], device=DEVICE)  # type: ignore
    return model
```

[18]:
```
def freeze_encoder_only(model: SentenceTransformer):
    # SentenceTransformer stores modules in model._modules (OrderedDict-like).␣
↪The transformer is index 0.
    # Simpler: freeze parameters in modules that are instances of models.
↪Transformer
    for module in model._modules.values():
        if isinstance(module, models.Transformer):
            for p in module.parameters():
                p.requires_grad = False
```

[ ]:
```
def train_pooler_then_finetune(model: SentenceTransformer, train_examples,␣
↪val_examples, out_dir: str):
    # Step A: train pooler only (encoder frozen)
    freeze_encoder_only(model)
    train_dataloader = torch.utils.data.DataLoader(train_examples,␣
↪batch_size=BATCH_SIZE, shuffle=True)
    # Use CosineSimilarityLoss for contrastive-style or MSELoss for regression
    # (STS)
    loss_fct = losses.MultipleNegativesRankingLoss(model)
    evaluator = evaluation.EmbeddingSimilarityEvaluator.from_input_examples(
        val_examples, name="sts-val"
    )  # note this benchmark compares against human-annotated similarity scores;
    # ABOBA: we can't self-annotate sim for Swift or Verma so we can't get
    # encoder error
    model.fit(
        train_objectives=[(train_dataloader, loss_fct)],
        evaluator=evaluator,
        epochs=EPOCHS_POOLER,
        warmup_steps=100,
        output_path=os.path.join(out_dir, "stepA_pooler_only"),
        optimizer_params={"lr": POOLER_LR},
    )
    # Step B: unfreeze encoder and finetune whole model
    uks_torch.unfreeze_model_weights(model)
    # Recreate dataloader (sentence-transformers expects InputExamples in an
```

```
    # in-memory list)
    train_dataloader = torch.utils.data.DataLoader(train_examples,␣
 ↪batch_size=BATCH_SIZE, shuffle=True)
    loss_fct2 = losses.MultipleNegativesRankingLoss(
        model
    )  # good objective for contrastive training (requires positive pairs)
    model.fit(
        train_objectives=[(train_dataloader, loss_fct2)],
        evaluator=evaluator,
        epochs=EPOCHS_FINETUNE,
        warmup_steps=100,
        output_path=os.path.join(out_dir, "stepB_finetune"),
        optimizer_params={"lr": FINETUNE_LR},
    )
    # -------------------------
```

[20]:
```
s_model = build_model(BASE_MODEL, TARGET_DIM)
```

```
config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]
vocab.txt: 0.00B [00:00, ?B/s]
tokenizer.json: 0.00B [00:00, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
```

[21]:
```
s_model
```

[21]:
```
SentenceTransformer(
  (0): Transformer({'max_seq_length': 128, 'do_lower_case': False,
'architecture': 'BertModel'})
  (1): Pooling({'word_embedding_dimension': 384, 'pooling_mode_cls_token':
False, 'pooling_mode_mean_tokens': True, 'pooling_mode_max_tokens': False,
'pooling_mode_mean_sqrt_len_tokens': False, 'pooling_mode_weightedmean_tokens':
False, 'pooling_mode_lasttoken': False, 'include_prompt': True})
  (2): Dense({'in_features': 384, 'out_features': 64, 'bias': True,
'activation_function': 'torch.nn.modules.activation.Tanh'})
)
```

[32]:
```
train_pooler_then_finetune(s_model, df_stsb_train, df_stsb_val, out_dir="output/
 ↪sentence_transformer_finetuned")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[32], line 1
```

```
----> 1␣
 ↪train_pooler_then_finetune(s_model, df_stsb_train, df_stsb_val, out_dir=

Cell In[19], line 8, in train_pooler_then_finetune(model, train_examples,␣
 ↪val_examples, out_dir)
      5 # Use CosineSimilarityLoss for contrastive-style or MSELoss for␣
 ↪regression
      6 # (STS)
      7 loss_fct = losses.CosineSimilarityLoss(model)
----> 8 evaluator = evaluation.EmbeddingSimilarityEvaluator.from_input_examples
      9     val_examples, name=
     10 )  # note this benchmark compares against human-annotated similarity␣
 ↪scores;
     11 # ABOBA: we can't self-annotate sim for Swift or Verma so we can't get
     12 # encoder error
     13 model.fit(
     14     train_objectives=[(train_dataloader, loss_fct)],
     15     evaluator=evaluator,
   (…)     19     optimizer_params={"lr": POOLER_LR},
     20 )

File ~/micromamba/envs/kret_312/lib/python3.12/site-packages/
 ↪sentence_transformers/evaluation/EmbeddingSimilarityEvaluator.py:148, in␣
 ↪EmbeddingSimilarityEvaluator.from_input_examples(cls, examples, **kwargs)
    145 scores = []
    147 for example in examples:
--> 148     sentences1.append(example.texts[0])
    149     sentences2.append(example.texts[1])
    150     scores.append(example.label)

AttributeError: 'str' object has no attribute 'texts'
```

[ ]: