S6001_train

1 Boilerplate

1.1 prod: Logger and Location

```
import nvm
import srsly
import pathlib
logZ = nvm.Log0(write=True, stream_lvl="INFO", file_lvl="DEBUG")
log0 = logZ.logger
locations = """
stardust7:
  jiko: cc/dev/c2023a/c0501_bertagent_devel/code/c0001_stage_04/
buka2:
  jiko: cc/dev/c2023a/c0501_bertagent_devel/code/c0001_stage_04/
"""
locations = srsly.yaml_loads(locations)
log0.info(f"{nvm.chdir(locations)}")
```

1.2 prod: Auto reload

```
get_ipython().run_line_magic("load_ext", "autoreload")
get_ipython().run_line_magic("autoreload", "2")
```

2 Imports

2.1 prod: NVM

```
from nvm import disp_df
from nvm import repr_df
from nvm import rdf
from nvm import ddf
from nvm import clean_str
from nvm.aux_str import CLEAN_STR_MAPPINGS_LARGE as maps0
from nvm.aux_str import REGEX_ABC_DASH_XYZ_ASTERISK as re0
from nvm.aux_pandas import fix_column_names
```

2.2 prod: Basics

```
import os
import pathlib
import numpy as np
import pandas as pd
import re
import json
import yaml
import srsly
import uuid
import random
import numbers
from collections import OrderedDict
from contextlib import ExitStack
import warnings
# warnings.warn("\nwarning")
from hashlib import md5
import humanfriendly as hf
import time
import datetime as dt
from pytz import timezone as tz
tz0 = tz("Europe/Berlin")
from glob import glob
from tqdm import tqdm
import logging
log0.info("DONE: basic imports")
```

2.3 prod: Extra imports and settings

```
from contexttimer import Timer
import textwrap

HOME = pathlib.Path.home()

tqdm.pandas()

import matplotlib
from matplotlib import pyplot as plt
# import seaborn as sns
# import plotly.graph_objects as go
# import plotly.express as px

# get_ipython().run_line_magic("matplotlib", "qt")
# get_ipython().run_line_magic("matplotlib", "inline")

with Timer() as elapsed:
    time.sleep(0.001)

log0.info(hf.format_timespan(elapsed.elapsed))

log0.info("DONE: extra imports and settings")
```

3 Extra Imports

3.1 prod: More extra imports and settings

```
import numpy as np
import pandas as pd
from sklearn.utils import Bunch
import torch
import itertools
import gc

from transformers import AutoModelForSequenceClassification
from transformers import AutoTokenizer
from helpers import BERTAgentSentencesPredictor
from helpers import compute_metrics
from helpers import json_serializable_or_str_repr
from helpers import train_bert_model

log0.info(f"{torch.cuda.is_available()}")
log0.info("DONE: more extra imports and settings")
```

4 Process

4.1 prod: Metadata

```
metadata = Bunch()

metadata.data_dir = "../../data/d0010_training-data/"
metadata.data_dir = pathlib.Path(metadata.data_dir)

metadata.goldstd = "gs0x"  # GOLD STANDARD
metadata.dataset = "tiny"  # TINY
metadata.dataset = "temp"  # TEMP
metadata.dataset = "ft4x"  # LEGACY
metadata.dataset = "ft3x"  # EVERYTHING
metadata.dataset = "ft2x"  # EXTENDED/AUGUMENTED
metadata.dataset = "ft1x"  # SIMPLE
metadata.dataset = "ft0x"  # EXAMPLES ONLY

metadata.extn = ".pkl"

metadata.testing = "both"
metadata.testing = "gold"

metadata.num_epochs = 12
metadata.batch_size = 64
metadata.random_state = 42
metadata.seed = metadata.random_state
metadata.max_length=128
metadata.status = "TRIALS"
metadata.status = "DEPLOY"
```

```
metadata.model_name = "bert-base-uncased"
metadata.model_name = "roberta-base"

log0.info(f"METADATA:\n{textwrap.indent(srsly.yaml_dumps(json_serializable_or_str_repr(dict(metadata
```

4.2 prod: Training in batches

```
METADATA = Bunch(**{key: [val] for key, val in metadata.items()})

METADATA.pop("dataset", None)
# METADATA.dataset = ["temp", "tiny"]
# METADATA.dataset = ["temp", "ft1x"]
METADATA.dataset = ["ft0x"]
METADATA.dataset = ["ft0x", "ft1x", "ft2x", "ft3x", "ft4x"]
METADATA.dataset = ["ft1x", "ft2x", "ft3x"]

METADATA.pop("model_name", None)
METADATA.model_name = ["bert-base-uncased", "roberta-base"]

METADATA = Bunch(**{key: sorted(list(set(val))) for key, val in METADATA.items()})

log0.info(f"METADATA:\n{textwrap.indent(srsly.yaml_dumps(json_serializable_or_str_repr(dict(METADATA

arg_names = list(METADATA.keys())
arg_values = list(METADATA.values())

for values in itertools.product(*arg_values):
    arg_dict = Bunch(**dict(zip(arg_names, values)))
    print(dict(**arg_dict))

print("=" * 77)
print(f"METADATA:\n{textwrap.indent(srsly.yaml_dumps(json_serializable_or_str_repr(dict(METADATA)))
print("=" * 77)

for values in itertools.product(*arg_values):
    arg_dict = Bunch(**dict(zip(arg_names, values)))
    print(dict(**arg_dict))

    trainer, ds0, summary, dir0 = train_bert_model(
        metadata=arg_dict,
        compute_metrics=compute_metrics,
        log0=log0,
    )
    try:
        del trainer, ds0, summary, dir0
    except NameError:
        pass

    gc.collect()
    torch.cuda.empty_cache()
```