"""

Training job for the heavy ranker of the push notification service.

"""

from datetime import datetime

import json

import os

import twml

from ..libs.metric\_fn\_utils import flip\_disliked\_labels, get\_metric\_fn

from ..libs.model\_utils import read\_config

from ..libs.warm\_start\_utils import get\_feature\_list\_for\_heavy\_ranking, warm\_start\_checkpoint

from .features import get\_feature\_config

from .model\_pools import ALL\_MODELS

from .params import load\_graph\_params

from .run\_args import get\_training\_arg\_parser

import tensorflow.compat.v1 as tf

from tensorflow.compat.v1 import logging

def main() -> None:

args, \_ = get\_training\_arg\_parser().parse\_known\_args()

logging.info(f"Parsed args: {args}")

params = load\_graph\_params(args)

logging.info(f"Loaded graph params: {params}")

param\_file = os.path.join(args.save\_dir, "params.json")

logging.info(f"Saving graph params to: {param\_file}")

with tf.io.gfile.GFile(param\_file, mode="w") as file:

json.dump(params.json(), file, ensure\_ascii=False, indent=4)

logging.info(f"Get Feature Config: {args.feature\_list}")

feature\_list = read\_config(args.feature\_list).items()

feature\_config = get\_feature\_config(

data\_spec\_path=args.data\_spec,

params=params,

feature\_list\_provided=feature\_list,

)

feature\_list\_path = args.feature\_list

warm\_start\_from = args.warm\_start\_from

if args.warm\_start\_base\_dir:

logging.info(f"Get warm started model from: {args.warm\_start\_base\_dir}.")

continuous\_binary\_feat\_list\_save\_path = os.path.join(

args.warm\_start\_base\_dir, "continuous\_binary\_feat\_list.json"

)

warm\_start\_folder = os.path.join(args.warm\_start\_base\_dir, "best\_checkpoint")

job\_name = os.path.basename(args.save\_dir)

ws\_output\_ckpt\_folder = os.path.join(args.warm\_start\_base\_dir, f"warm\_start\_for\_{job\_name}")

if tf.io.gfile.exists(ws\_output\_ckpt\_folder):

tf.io.gfile.rmtree(ws\_output\_ckpt\_folder)

tf.io.gfile.mkdir(ws\_output\_ckpt\_folder)

warm\_start\_from = warm\_start\_checkpoint(

warm\_start\_folder,

continuous\_binary\_feat\_list\_save\_path,

feature\_list\_path,

args.data\_spec,

ws\_output\_ckpt\_folder,

)

logging.info(f"Created warm\_start\_from\_ckpt {warm\_start\_from}.")

logging.info("Build Trainer.")

metric\_fn = get\_metric\_fn("OONC\_Engagement" if len(params.tasks) == 2 else "OONC", False)

trainer = twml.trainers.DataRecordTrainer(

name="magic\_recs",

params=args,

build\_graph\_fn=lambda \*args: ALL\_MODELS[params.model.name](params=params)(\*args),

save\_dir=args.save\_dir,

run\_config=None,

feature\_config=feature\_config,

metric\_fn=flip\_disliked\_labels(metric\_fn),

warm\_start\_from=warm\_start\_from,

)

logging.info("Build train and eval input functions.")

train\_input\_fn = trainer.get\_train\_input\_fn(shuffle=True)

eval\_input\_fn = trainer.get\_eval\_input\_fn(repeat=False, shuffle=False)

learn = trainer.learn

if args.distributed or args.num\_workers is not None:

learn = trainer.train\_and\_evaluate

if not args.directly\_export\_best:

logging.info("Starting training")

start = datetime.now()

learn(

early\_stop\_minimize=False,

early\_stop\_metric="pr\_auc\_unweighted\_OONC",

early\_stop\_patience=args.early\_stop\_patience,

early\_stop\_tolerance=args.early\_stop\_tolerance,

eval\_input\_fn=eval\_input\_fn,

train\_input\_fn=train\_input\_fn,

)

logging.info(f"Total training time: {datetime.now() - start}")

else:

logging.info("Directly exporting the model")

if not args.export\_dir:

args.export\_dir = os.path.join(args.save\_dir, "exported\_models")

logging.info(f"Exporting the model to {args.export\_dir}.")

start = datetime.now()

twml.contrib.export.export\_fn.export\_all\_models(

trainer=trainer,

export\_dir=args.export\_dir,

parse\_fn=feature\_config.get\_parse\_fn(),

serving\_input\_receiver\_fn=feature\_config.get\_serving\_input\_receiver\_fn(),

export\_output\_fn=twml.export\_output\_fns.batch\_prediction\_continuous\_output\_fn,

)

logging.info(f"Total model export time: {datetime.now() - start}")

logging.info(f"The MLP directory is: {args.save\_dir}")

continuous\_binary\_feat\_list\_save\_path = os.path.join(

args.save\_dir, "continuous\_binary\_feat\_list.json"

)

logging.info(

f"Saving the list of continuous and binary features to {continuous\_binary\_feat\_list\_save\_path}."

)

continuous\_binary\_feat\_list = get\_feature\_list\_for\_heavy\_ranking(

feature\_list\_path, args.data\_spec

)

twml.util.write\_file(

continuous\_binary\_feat\_list\_save\_path, continuous\_binary\_feat\_list, encode="json"

)

if \_\_name\_\_ == "\_\_main\_\_":

main()

logging.info("Done.")