from datetime import datetime

from functools import partial

import os

from ..libs.group\_metrics import (

run\_group\_metrics\_light\_ranking,

run\_group\_metrics\_light\_ranking\_in\_bq,

)

from ..libs.metric\_fn\_utils import get\_metric\_fn

from ..libs.model\_args import get\_arg\_parser\_light\_ranking

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

from .deep\_norm import build\_graph, DataRecordTrainer, get\_config\_func, logging

# checkstyle: noqa

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

parser = get\_arg\_parser\_light\_ranking()

parser.add\_argument(

"--eval\_checkpoint",

default=None,

type=str,

help="Which checkpoint to use for evaluation",

)

parser.add\_argument(

"--saved\_model\_path",

default=None,

type=str,

help="Path to saved model for evaluation",

)

parser.add\_argument(

"--run\_binary\_metrics",

default=False,

action="store\_true",

help="Whether to compute the basic binary metrics for Light Ranking.",

)

opt = parser.parse\_args()

logging.info("parse is: ")

logging.info(opt)

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

feature\_config = get\_config\_func(opt.feat\_config\_type)(

data\_spec\_path=opt.data\_spec,

feature\_list\_provided=feature\_list,

opt=opt,

add\_gbdt=opt.use\_gbdt\_features,

run\_light\_ranking\_group\_metrics\_in\_bq=opt.run\_light\_ranking\_group\_metrics\_in\_bq,

)

# -----------------------------------------------

# Create Trainer

# -----------------------------------------------

trainer = DataRecordTrainer(

name=opt.model\_trainer\_name,

params=opt,

build\_graph\_fn=partial(build\_graph, run\_light\_ranking\_group\_metrics\_in\_bq=True),

save\_dir=opt.save\_dir,

run\_config=None,

feature\_config=feature\_config,

metric\_fn=get\_metric\_fn(opt.task\_name, use\_stratify\_metrics=False),

)

# -----------------------------------------------

# Model Evaluation

# -----------------------------------------------

logging.info("Evaluating...")

start = datetime.now()

if opt.run\_binary\_metrics:

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

eval\_steps = None if (opt.eval\_steps is not None and opt.eval\_steps < 0) else opt.eval\_steps

trainer.estimator.evaluate(eval\_input\_fn, steps=eval\_steps, checkpoint\_path=opt.eval\_checkpoint)

if opt.run\_light\_ranking\_group\_metrics\_in\_bq:

run\_group\_metrics\_light\_ranking\_in\_bq(

trainer=trainer, params=opt, checkpoint\_path=opt.eval\_checkpoint

)

if opt.run\_light\_ranking\_group\_metrics:

run\_group\_metrics\_light\_ranking(

trainer=trainer,

data\_dir=os.path.join(opt.eval\_data\_dir, opt.eval\_start\_datetime),

model\_path=opt.saved\_model\_path,

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

)

end = datetime.now()

logging.info("Evaluating time: " + str(end - start))