import enum

import json

from typing import List, Optional

from .lib.params import BlockParams, ClemNetParams, ConvParams, DenseParams, TopLayerParams

from pydantic import BaseModel, Extra, NonNegativeFloat

import tensorflow.compat.v1 as tf

# checkstyle: noqa

class ExtendedBaseModel(BaseModel):

class Config:

extra = Extra.forbid

class SparseFeaturesParams(ExtendedBaseModel):

bits: int

embedding\_size: int

class FeaturesParams(ExtendedBaseModel):

sparse\_features: Optional[SparseFeaturesParams]

class ModelTypeEnum(str, enum.Enum):

clemnet: str = "clemnet"

class ModelParams(ExtendedBaseModel):

name: ModelTypeEnum

features: FeaturesParams

architecture: ClemNetParams

class TaskNameEnum(str, enum.Enum):

oonc: str = "OONC"

engagement: str = "Engagement"

class Task(ExtendedBaseModel):

name: TaskNameEnum

label: str

score\_weight: NonNegativeFloat

DEFAULT\_TASKS = [

Task(name=TaskNameEnum.oonc, label="label", score\_weight=0.9),

Task(name=TaskNameEnum.engagement, label="label.engagement", score\_weight=0.1),

]

class GraphParams(ExtendedBaseModel):

tasks: List[Task] = DEFAULT\_TASKS

model: ModelParams

weight: Optional[str]

DEFAULT\_ARCHITECTURE\_PARAMS = ClemNetParams(

blocks=[

BlockParams(

activation="relu",

conv=ConvParams(kernel\_size=3, filters=5),

dense=DenseParams(output\_size=output\_size),

residual=False,

)

for output\_size in [1024, 512, 256, 128]

],

top=TopLayerParams(n\_labels=1),

)

DEFAULT\_GRAPH\_PARAMS = GraphParams(

model=ModelParams(

name=ModelTypeEnum.clemnet,

architecture=DEFAULT\_ARCHITECTURE\_PARAMS,

features=FeaturesParams(sparse\_features=SparseFeaturesParams(bits=18, embedding\_size=50)),

),

)

def load\_graph\_params(args) -> GraphParams:

params = DEFAULT\_GRAPH\_PARAMS

if args.param\_file:

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

params = GraphParams.parse\_obj(json.load(file))

return params