package com.twitter.product\_mixer.component\_library.scorer.tensorbuilder

import com.twitter.ml.api.thriftscala.FloatTensor

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.FeatureWithDefaultOnFailure

import com.twitter.product\_mixer.core.feature.ModelFeatureName

import com.twitter.product\_mixer.core.feature.featuremap.featurestorev1.FeatureStoreV1FeatureMap.\_

import com.twitter.product\_mixer.core.feature.featurestorev1.FeatureStoreV1CandidateFeature

import com.twitter.product\_mixer.core.feature.featurestorev1.FeatureStoreV1QueryFeature

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import inference.GrpcService.ModelInferRequest.InferInputTensor

class CandidateInferInputTensorBuilder[-Candidate <: UniversalNoun[Any], +Value](

builder: InferInputTensorBuilder[Value],

features: Set[\_ <: Feature[Candidate, \_] with ModelFeatureName]) {

def apply(

candidates: Seq[CandidateWithFeatures[Candidate]],

): Seq[InferInputTensor] = {

features.flatMap { feature =>

val featureValues: Seq[Value] = feature match {

case feature: FeatureStoreV1CandidateFeature[\_, Candidate, \_, Value] =>

candidates.map(\_.features.getFeatureStoreV1CandidateFeature(feature))

case feature: FeatureStoreV1QueryFeature[\_, \_, \_] =>

throw new UnexpectedFeatureTypeException(feature)

case feature: FeatureWithDefaultOnFailure[Candidate, Value] =>

candidates.map(\_.features.getTry(feature).toOption.getOrElse(feature.defaultValue))

case feature: Feature[Candidate, Value] =>

candidates.map(\_.features.get(feature))

}

builder.apply(feature.featureName, featureValues)

}.toSeq

}

}

case class CandidateBooleanInferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, Boolean] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, Boolean](

BooleanInferInputTensorBuilder,

features)

case class CandidateBytesInferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, String] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, String](

BytesInferInputTensorBuilder,

features)

case class CandidateFloat32InferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, \_ <: AnyVal] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, AnyVal](

Float32InferInputTensorBuilder,

features)

case class CandidateFloatTensorInferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, FloatTensor] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, FloatTensor](

FloatTensorInferInputTensorBuilder,

features)

case class CandidateInt64InferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, \_ <: AnyVal] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, AnyVal](

Int64InferInputTensorBuilder,

features)

case class CandidateSparseMapInferInputTensorBuilder[-Candidate <: UniversalNoun[Any]](

features: Set[\_ <: Feature[Candidate, Option[Map[Int, Double]]] with ModelFeatureName])

extends CandidateInferInputTensorBuilder[Candidate, Option[Map[Int, Double]]](

SparseMapInferInputTensorBuilder,

features)