package com.twitter.ann.featurestore

import com.twitter.ann.common.EmbeddingProducer

import com.twitter.finagle.stats.{InMemoryStatsReceiver, StatsReceiver}

import com.twitter.ml.api.embedding.{Embedding, EmbeddingSerDe}

import com.twitter.ml.api.thriftscala

import com.twitter.ml.api.thriftscala.{Embedding => TEmbedding}

import com.twitter.ml.featurestore.lib.dataset.online.VersionedOnlineAccessDataset

import com.twitter.ml.featurestore.lib.{EntityId, RawFloatTensor}

import com.twitter.ml.featurestore.lib.dataset.DatasetParams

import com.twitter.ml.featurestore.lib.entity.EntityWithId

import com.twitter.ml.featurestore.lib.feature.{BoundFeature, BoundFeatureSet}

import com.twitter.ml.featurestore.lib.online.{FeatureStoreClient, FeatureStoreRequest}

import com.twitter.ml.featurestore.lib.params.FeatureStoreParams

import com.twitter.stitch.Stitch

import com.twitter.strato.opcontext.Attribution

import com.twitter.strato.client.Client

object FeatureStoreEmbeddingProducer {

def apply[T <: EntityId](

dataset: VersionedOnlineAccessDataset[T, TEmbedding],

version: Long,

boundFeature: BoundFeature[T, RawFloatTensor],

client: Client,

statsReceiver: StatsReceiver = new InMemoryStatsReceiver,

featureStoreAttributions: Seq[Attribution] = Seq.empty

): EmbeddingProducer[EntityWithId[T]] = {

val featureStoreParams = FeatureStoreParams(

perDataset = Map(

dataset.id -> DatasetParams(datasetVersion = Some(version))

),

global = DatasetParams(attributions = featureStoreAttributions)

)

val featureStoreClient = FeatureStoreClient(

BoundFeatureSet(boundFeature),

client,

statsReceiver,

featureStoreParams

)

new FeatureStoreEmbeddingProducer(boundFeature, featureStoreClient)

}

}

private[featurestore] class FeatureStoreEmbeddingProducer[T <: EntityId](

boundFeature: BoundFeature[T, RawFloatTensor],

featureStoreClient: FeatureStoreClient)

extends EmbeddingProducer[EntityWithId[T]] {

// Looks up embedding from online feature store for an entity.

override def produceEmbedding(input: EntityWithId[T]): Stitch[Option[Embedding[Float]]] = {

val featureStoreRequest = FeatureStoreRequest(

entityIds = Seq(input)

)

Stitch.callFuture(featureStoreClient(featureStoreRequest).map { predictionRecord =>

predictionRecord.getFeatureValue(boundFeature) match {

case Some(featureValue) => {

val embedding = EmbeddingSerDe.floatEmbeddingSerDe.fromThrift(

thriftscala.Embedding(Some(featureValue.value))

)

Some(embedding)

}

case \_ => None

}

})

}

}