package com.twitter.product\_mixer.component\_library.scorer.tweet\_tlx

import com.twitter.product\_mixer.component\_library.model.candidate.TweetCandidate

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

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

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.scorer.Scorer

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

import com.twitter.product\_mixer.core.model.common.identifier.ScorerIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.stitch.Stitch

import com.twitter.timelinescorer.thriftscala.v1

import com.twitter.timelinescorer.{thriftscala => t}

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* @note This Feature is shared with

\* [[com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_tlx.TweetTLXScoreCandidateFeatureHydrator]]

\* and

\* [[com.twitter.product\_mixer.component\_library.scorer.tweet\_tlx.TweetTLXStratoScorer]]

\* as the these components should not be used at the same time by the same Product

\*/

object TLXScore extends FeatureWithDefaultOnFailure[TweetCandidate, Option[Double]] {

override val defaultValue = None

}

/\*\*

\* Score Tweets via Timeline Scorer (TLX) Thrift API

\*

\* @note This is the [[Scorer]] version of

\* [[com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_tlx.TweetTLXScoreCandidateFeatureHydrator]]

\*/

@Singleton

class TweetTLXThriftScorer @Inject() (timelineScorerClient: t.TimelineScorer.MethodPerEndpoint)

extends Scorer[PipelineQuery, TweetCandidate] {

override val identifier: ScorerIdentifier = ScorerIdentifier("TLX")

override val features: Set[Feature[\_, \_]] = Set(TLXScore)

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = {

val userId = query.getOptionalUserId

val tweetScoringQuery = v1.TweetScoringQuery(

predictionPipeline = v1.PredictionPipeline.Recap,

tweetIds = candidates.map(\_.candidate.id))

val tweetScoringRequest = t.TweetScoringRequest.V1(

v1.TweetScoringRequest(

tweetScoringRequestContext = Some(v1.TweetScoringRequestContext(userId = userId)),

tweetScoringQueries = Some(Seq(tweetScoringQuery)),

retrieveFeatures = Some(false)

))

Stitch.callFuture(timelineScorerClient.getTweetScores(tweetScoringRequest)).map {

case t.TweetScoringResponse.V1(response) =>

val tweetIdScoreMap = response.tweetScoringResults

.flatMap {

\_.headOption.map {

\_.scoredTweets.flatMap(tweet => tweet.tweetId.map(\_ -> tweet.score))

}

}.getOrElse(Seq.empty).toMap

candidates.map { candidateWithFeatures =>

val score = tweetIdScoreMap.getOrElse(candidateWithFeatures.candidate.id, None)

FeatureMapBuilder()

.add(TLXScore, score)

.build()

}

case t.TweetScoringResponse.UnknownUnionField(field) =>

throw new UnsupportedOperationException(s"Unknown response type: ${field.field.name}")

}

}

}