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

import com.twitter.ml.featurestore.timelines.thriftscala.TimelineScorerScoreView

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.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.strato.catalog.Fetch.Result

import com.twitter.strato.generated.client.ml.featureStore.TimelineScorerTweetScoresV1ClientColumn

import com.twitter.timelinescorer.thriftscala.v1

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

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

\*

\* @note This results in an additional hop through Strato Server

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

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

\*/

@Singleton

class TweetTLXStratoScorer @Inject() (column: TimelineScorerTweetScoresV1ClientColumn)

extends Scorer[PipelineQuery, TweetCandidate] {

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

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

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = query.getOptionalUserId match {

case Some(userId) => getScoredTweetsFromTLX(userId, candidates.map(\_.candidate))

case \_ =>

val defaultFeatureMap = FeatureMapBuilder().add(TLXScore, None).build()

Stitch.value(candidates.map(\_ => defaultFeatureMap))

}

def getScoredTweetsFromTLX(

userId: Long,

tweetCandidates: Seq[TweetCandidate]

): Stitch[Seq[FeatureMap]] = Stitch.collect(tweetCandidates.map { candidate =>

column.fetcher

.fetch(candidate.id, TimelineScorerScoreView(Some(userId)))

.map {

case Result(Some(v1.ScoredTweet(\_, score, \_, \_)), \_) =>

FeatureMapBuilder()

.add(TLXScore, score)

.build()

case fetchResult => throw new Exception(s"Invalid response from TLX: ${fetchResult.v}")

}

})

}