package com.twitter.product\_mixer.component\_library.candidate\_source.timeline\_ranker

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

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

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSourceWithExtractedFeatures

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidatesWithSourceFeatures

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

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

import com.twitter.stitch.Stitch

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

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Map of tweetId -> sourceTweet of retweets present in Timeline Ranker candidates list.

\* These tweets are used only for further ranking. They are not returned to the end user.

\*/

object TimelineRankerInNetworkSourceTweetsByTweetIdMapFeature

extends Feature[PipelineQuery, Map[Long, t.CandidateTweet]]

@Singleton

class TimelineRankerInNetworkCandidateSource @Inject() (

timelineRankerClient: t.TimelineRanker.MethodPerEndpoint)

extends CandidateSourceWithExtractedFeatures[t.RecapQuery, t.CandidateTweet] {

override val identifier: CandidateSourceIdentifier =

CandidateSourceIdentifier("TimelineRankerInNetwork")

override def apply(

request: t.RecapQuery

): Stitch[CandidatesWithSourceFeatures[t.CandidateTweet]] = {

Stitch

.callFuture(timelineRankerClient.getRecycledTweetCandidates(Seq(request)))

.map { response: Seq[t.GetCandidateTweetsResponse] =>

val candidates =

response.headOption.flatMap(\_.candidates).getOrElse(Seq.empty).filter(\_.tweet.nonEmpty)

val sourceTweetsByTweetId =

response.headOption

.flatMap(\_.sourceTweets).getOrElse(Seq.empty).filter(\_.tweet.nonEmpty)

.map { candidate =>

(candidate.tweet.get.id, candidate)

}.toMap

val sourceTweetsByTweetIdMapFeature = FeatureMapBuilder()

.add(TimelineRankerInNetworkSourceTweetsByTweetIdMapFeature, sourceTweetsByTweetId)

.build()

CandidatesWithSourceFeatures(

candidates = candidates,

features = sourceTweetsByTweetIdMapFeature)

}

}

}