package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator

import com.twitter.contentrecommender.{thriftscala => cr}

import com.twitter.home\_mixer.model.HomeFeatures.CandidateSourceIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.TSPMetricTagFeature

import com.twitter.home\_mixer.model.HomeFeatures.TopicContextFunctionalityTypeFeature

import com.twitter.home\_mixer.model.HomeFeatures.TopicIdSocialContextFeature

import com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.adapters.inferred\_topic.InferredTopicAdapter

import com.twitter.ml.api.DataRecord

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.datarecord.DataRecordInAFeature

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.feature\_hydrator.BulkCandidateFeatureHydrator

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.BasicTopicContextFunctionalityType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.RecommendationTopicContextFunctionalityType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.TopicContextFunctionalityType

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

import com.twitter.product\_mixer.core.util.OffloadFuturePools

import com.twitter.stitch.Stitch

import com.twitter.timelines.clients.strato.topics.TopicSocialProofClient

import com.twitter.timelineservice.suggests.logging.candidate\_tweet\_source\_id.{thriftscala => sid}

import com.twitter.topiclisting.TopicListingViewerContext

import com.twitter.tsp.{thriftscala => tsp}

import javax.inject.Inject

import javax.inject.Singleton

import scala.collection.JavaConverters.\_

object TSPInferredTopicFeature extends Feature[TweetCandidate, Map[Long, Double]]

object TSPInferredTopicDataRecordFeature

extends DataRecordInAFeature[TweetCandidate]

with FeatureWithDefaultOnFailure[TweetCandidate, DataRecord] {

override def defaultValue: DataRecord = new DataRecord()

}

@Singleton

class TSPInferredTopicFeatureHydrator @Inject() (

topicSocialProofClient: TopicSocialProofClient)

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate] {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("TSPInferredTopic")

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

TSPInferredTopicFeature,

TSPInferredTopicDataRecordFeature,

TopicIdSocialContextFeature,

TopicContextFunctionalityTypeFeature

)

private val topK = 3

private val SourcesToSetSocialProof: Set[sid.CandidateTweetSourceId] =

Set(sid.CandidateTweetSourceId.Simcluster)

private val DefaultFeatureMap = FeatureMapBuilder()

.add(TSPInferredTopicFeature, Map.empty[Long, Double])

.add(TSPInferredTopicDataRecordFeature, new DataRecord())

.add(TopicIdSocialContextFeature, None)

.add(TopicContextFunctionalityTypeFeature, None)

.build()

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = OffloadFuturePools.offloadFuture {

val tags = candidates.collect {

case candidate if candidate.features.getTry(TSPMetricTagFeature).isReturn =>

candidate.candidate.id -> candidate.features

.getOrElse(TSPMetricTagFeature, Set.empty[tsp.MetricTag])

}.toMap

val topicSocialProofRequest = tsp.TopicSocialProofRequest(

userId = query.getRequiredUserId,

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

displayLocation = cr.DisplayLocation.HomeTimeline,

topicListingSetting = tsp.TopicListingSetting.Followable,

context = TopicListingViewerContext.fromClientContext(query.clientContext).toThrift,

bypassModes = None,

// Only TweetMixer source has this data. Convert the TweetMixer metric tag to tsp metric tag.

tags = if (tags.isEmpty) None else Some(tags)

)

topicSocialProofClient

.getTopicTweetSocialProofResponse(topicSocialProofRequest)

.map {

case Some(response) =>

handleResponse(response, candidates)

case \_ => candidates.map { \_ => DefaultFeatureMap }

}

}

private def handleResponse(

response: tsp.TopicSocialProofResponse,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Seq[FeatureMap] = {

candidates.map { candidate =>

val topicWithScores = response.socialProofs.getOrElse(candidate.candidate.id, Seq.empty)

if (topicWithScores.nonEmpty) {

val (socialProofId, socialProofFunctionalityType) =

if (candidate.features

.getOrElse(CandidateSourceIdFeature, None)

.exists(SourcesToSetSocialProof.contains)) {

getSocialProof(topicWithScores)

} else (None, None)

val inferredTopicFeatures =

topicWithScores.sortBy(-\_.score).take(topK).map(a => (a.topicId, a.score)).toMap

val inferredTopicDataRecord =

InferredTopicAdapter.adaptToDataRecords(inferredTopicFeatures).asScala.head

FeatureMapBuilder()

.add(TSPInferredTopicFeature, inferredTopicFeatures)

.add(TSPInferredTopicDataRecordFeature, inferredTopicDataRecord)

.add(TopicIdSocialContextFeature, socialProofId)

.add(TopicContextFunctionalityTypeFeature, socialProofFunctionalityType)

.build()

} else DefaultFeatureMap

}

}

private def getSocialProof(

topicWithScores: Seq[tsp.TopicWithScore]

): (Option[Long], Option[TopicContextFunctionalityType]) = {

val followingTopicId = topicWithScores.collectFirst {

case tsp.TopicWithScore(topicId, \_, \_, Some(tsp.TopicFollowType.Following)) => topicId

}

if (followingTopicId.nonEmpty) {

return (followingTopicId, Some(BasicTopicContextFunctionalityType))

}

val implicitFollowingId = topicWithScores.collectFirst {

case tsp.TopicWithScore(topicId, \_, \_, Some(tsp.TopicFollowType.ImplicitFollow)) =>

topicId

}

if (implicitFollowingId.nonEmpty) {

return (implicitFollowingId, Some(RecommendationTopicContextFunctionalityType))

}

(None, None)

}

}