package com.twitter.cr\_mixer.similarity\_engine

import com.twitter.cr\_mixer.model.TweetWithAuthor

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.search.earlybird.thriftscala.EarlybirdRequest

import com.twitter.search.earlybird.thriftscala.EarlybirdResponseCode

import com.twitter.search.earlybird.thriftscala.EarlybirdService

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

/\*\*

\* This trait is a base trait for Earlybird similarity engines. All Earlybird similarity

\* engines extend from it and override the construction method for EarlybirdRequest

\*/

trait EarlybirdSimilarityEngineBase[EarlybirdSearchQuery]

extends ReadableStore[EarlybirdSearchQuery, Seq[TweetWithAuthor]] {

def earlybirdSearchClient: EarlybirdService.MethodPerEndpoint

def statsReceiver: StatsReceiver

def getEarlybirdRequest(query: EarlybirdSearchQuery): Option[EarlybirdRequest]

override def get(query: EarlybirdSearchQuery): Future[Option[Seq[TweetWithAuthor]]] = {

getEarlybirdRequest(query)

.map { earlybirdRequest =>

earlybirdSearchClient

.search(earlybirdRequest).map { response =>

response.responseCode match {

case EarlybirdResponseCode.Success =>

val earlybirdSearchResult =

response.searchResults

.map(

\_.results

.map(searchResult =>

TweetWithAuthor(

searchResult.id,

// fromUserId should be there since MetadataOptions.getFromUserId = true

searchResult.metadata.map(\_.fromUserId).getOrElse(0))).toSeq)

statsReceiver.scope("result").stat("size").add(earlybirdSearchResult.size)

earlybirdSearchResult

case e =>

statsReceiver.scope("failures").counter(e.getClass.getSimpleName).incr()

Some(Seq.empty)

}

}

}.getOrElse(Future.None)

}

}

object EarlybirdSimilarityEngineBase {

trait EarlybirdSearchQuery {

def seedUserIds: Seq[UserId]

def maxNumTweets: Int

}

}