package com.twitter.simclusters\_v2.candidate\_source

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateSource

import com.twitter.simclusters\_v2.candidate\_source.SimClustersANNCandidateSource.LookbackMediaTweetConfig

import com.twitter.simclusters\_v2.candidate\_source.SimClustersANNCandidateSource.SimClustersTweetCandidate

import com.twitter.util.Future

/\*\*

\* An abstraction layer that implements a lambda structure for ANNCandidate source.

\* Allows us to call an online store as well as an offline store from a single query.

\*/

case class SimClustersANNWrapperCandidateSource(

onlineANNSource: CandidateSource[SimClustersANNCandidateSource.Query, SimClustersTweetCandidate],

lookbackANNSource: CandidateSource[

SimClustersANNCandidateSource.Query,

SimClustersTweetCandidate

],

)(

statsReceiver: StatsReceiver)

extends CandidateSource[SimClustersANNCandidateSource.Query, SimClustersTweetCandidate] {

override def get(

query: SimClustersANNCandidateSource.Query

): Future[Option[Seq[SimClustersTweetCandidate]]] = {

val enableLookbackSource =

query.overrideConfig.exists(\_.enableLookbackSource.getOrElse(false))

val embeddingType = query.sourceEmbeddingId.embeddingType

val lookbackCandidatesFut =

if (enableLookbackSource &&

LookbackMediaTweetConfig.contains(embeddingType)) {

statsReceiver

.counter("lookback\_source", embeddingType.toString, "enable").incr()

statsReceiver.counter("lookback\_source", "enable").incr()

lookbackANNSource.get(query)

} else {

statsReceiver

.counter("lookback\_source", embeddingType.toString, "disable").incr()

Future.None

}

Future.join(onlineANNSource.get(query), lookbackCandidatesFut).map {

case (onlineCandidates, lookbackCandidates) =>

Some(

onlineCandidates.getOrElse(Nil) ++ lookbackCandidates.getOrElse(Nil)

)

}

}

override def name: String = this.getClass.getCanonicalName

}