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

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

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

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.util.Duration

import com.twitter.util.Time

object FilterUtil {

/\*\* Returns a list of tweets that are generated less than `maxTweetAgeHours` hours ago \*/

def tweetAgeFilter(

candidates: Seq[TweetWithScore],

maxTweetAgeHours: Duration

): Seq[TweetWithScore] = {

// Tweet IDs are approximately chronological (see http://go/snowflake),

// so we are building the earliest tweet id once

// The per-candidate logic here then be candidate.tweetId > earliestPermittedTweetId, which is far cheaper.

// See @cyao's phab on CrMixer generic age filter for reference https://phabricator.twitter.biz/D903188

val earliestTweetId = SnowflakeId.firstIdFor(Time.now - maxTweetAgeHours)

candidates.filter { candidate => candidate.tweetId >= earliestTweetId }

}

/\*\* Returns a list of tweet sources that are generated less than `maxTweetAgeHours` hours ago \*/

def tweetSourceAgeFilter(

candidates: Seq[SourceInfo],

maxTweetSignalAgeHoursParam: Duration

): Seq[SourceInfo] = {

// Tweet IDs are approximately chronological (see http://go/snowflake),

// so we are building the earliest tweet id once

// This filter applies to source signals. Some candidate source calls can be avoided if source signals

// can be filtered.

val earliestTweetId = SnowflakeId.firstIdFor(Time.now - maxTweetSignalAgeHoursParam)

candidates.filter { candidate =>

candidate.internalId match {

case InternalId.TweetId(tweetId) => tweetId >= earliestTweetId

case \_ => false

}

}

}

}