package com.twitter.product\_mixer.component\_library.filter.tweet\_impression

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.impressed\_tweets.ImpressedTweets

import com.twitter.product\_mixer.component\_library.model.candidate.BaseTweetCandidate

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.filter.FilterResult

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

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

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

import com.twitter.stitch.Stitch

/\*\*

\* Filters out tweets that the user has seen

\*/

case class TweetImpressionFilter[Candidate <: BaseTweetCandidate](

) extends Filter[PipelineQuery, Candidate] {

override val identifier: FilterIdentifier = FilterIdentifier("TweetImpression")

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

// Set of Tweets that have impressed the user

val impressedTweetsSet: Set[Long] = query.features match {

case Some(featureMap) => featureMap.getOrElse(ImpressedTweets, Seq.empty).toSet

case None => Set.empty

}

val (keptCandidates, removedCandidates) = candidates.partition { filteredCandidate =>

!impressedTweetsSet.contains(filteredCandidate.candidate.id)

}

Stitch.value(FilterResult(keptCandidates.map(\_.candidate), removedCandidates.map(\_.candidate)))

}

}