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

import com.twitter.product\_mixer.component\_library.feature\_hydrator.candidate.tweet\_tweetypie.IsReplyFeature

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 is a reply to a tweet

\*/

case class TweetIsNotReplyFilter[Candidate <: BaseTweetCandidate]()

extends Filter[PipelineQuery, Candidate] {

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

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val (kept, removed) = candidates

.partition { candidate =>

!candidate.features.get(IsReplyFeature)

}

val filterResult = FilterResult(

kept = kept.map(\_.candidate),

removed = removed.map(\_.candidate)

)

Stitch.value(filterResult)

}

}