package com.twitter.home\_mixer.product.for\_you.filter

import com.twitter.home\_mixer.model.HomeFeatures.TweetTextFeature

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

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

object TweetPreviewTextFilter extends Filter[PipelineQuery, TweetCandidate] {

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

private val PreviewTextLength = 50

private val MinTweetLength = PreviewTextLength \* 2

private val MaxNewlines = 2

private val HttpPrefix = "http://"

private val HttpsPrefix = "https://"

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[FilterResult[TweetCandidate]] = {

val (kept, removed) = candidates

.partition { candidate =>

val text = candidate.features.get(TweetTextFeature).getOrElse("")

text.length > MinTweetLength &&

text.take(PreviewTextLength).count(\_ == '\n') <= MaxNewlines &&

!(text.startsWith(HttpPrefix) || text.startsWith(HttpsPrefix))

}

val filterResult = FilterResult(

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

removed = removed.map(\_.candidate)

)

Stitch.value(filterResult)

}

}