package com.twitter.representationscorer.columns

import com.twitter.representationscorer.common.TweetId

import com.twitter.representationscorer.common.UserId

import com.twitter.representationscorer.thriftscala.SimClustersRecentEngagementSimilarities

import com.twitter.representationscorer.twistlyfeatures.Scorer

import com.twitter.stitch

import com.twitter.stitch.Stitch

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.ContactInfo

import com.twitter.strato.config.Policy

import com.twitter.strato.data.Conv

import com.twitter.strato.data.Description.PlainText

import com.twitter.strato.data.Lifecycle

import com.twitter.strato.fed.\_

import com.twitter.strato.thrift.ScroogeConv

import javax.inject.Inject

class SimClustersRecentEngagementSimilarityUserTweetEdgeColumn @Inject() (scorer: Scorer)

extends StratoFed.Column(

"recommendations/representation\_scorer/simClustersRecentEngagementSimilarity.UserTweetEdge")

with StratoFed.Fetch.Stitch {

override val policy: Policy = Common.rsxReadPolicy

override type Key = (UserId, TweetId)

override type View = Unit

override type Value = SimClustersRecentEngagementSimilarities

override val keyConv: Conv[Key] = Conv.ofType[(Long, Long)]

override val viewConv: Conv[View] = Conv.ofType

override val valueConv: Conv[Value] =

ScroogeConv.fromStruct[SimClustersRecentEngagementSimilarities]

override val contactInfo: ContactInfo = Info.contactInfo

override val metadata: OpMetadata = OpMetadata(

lifecycle = Some(Lifecycle.Production),

description = Some(

PlainText(

"User-Tweet scores based on the user's recent engagements"

))

)

override def fetch(key: Key, view: View): Stitch[Result[Value]] =

scorer

.get(key.\_1, key.\_2)

.map(found(\_))

.handle {

case stitch.NotFound => missing

}

}