package com.twitter.simclusters\_v2.scalding.mbcg

import com.google.common.collect.ImmutableSet

import com.twitter.dal.personal\_data.thriftjava.PersonalDataType.\_

import com.twitter.ml.api.DataType

import com.twitter.ml.api.Feature

import com.twitter.ml.api.Feature.SparseContinuous

import com.twitter.ml.api.Feature.Tensor

import com.twitter.ml.api.FeatureContext

import com.twitter.ml.api.constant.SharedFeatures

import java.util.{Map => JMap}

/\*

Features used for model-based candidate generation

\*/

object TweetAllFeatures {

val tweetId = SharedFeatures.TWEET\_ID

val tweetSimclusters =

new SparseContinuous(

"tweet.simcluster.log\_fav\_based\_embedding.20m\_145k\_2020",

ImmutableSet.of(InferredInterests))

.asInstanceOf[Feature[JMap[String, Double]]]

val authorF2vProducerEmbedding =

new Tensor(

"tweet.author\_follow2vec.producer\_embedding\_200",

DataType.FLOAT

)

private val allFeatures: Seq[Feature[\_]] = Seq(

tweetId,

tweetSimclusters,

authorF2vProducerEmbedding

)

val featureContext = new FeatureContext(allFeatures: \_\*)

}

object UserAllFeatures {

val userId = SharedFeatures.USER\_ID

val userSimclusters =

new SparseContinuous(

"user.iiape.log\_fav\_based\_embedding.20m\_145k\_2020",

ImmutableSet.of(InferredInterests))

.asInstanceOf[Feature[JMap[String, Double]]]

val userF2vConsumerEmbedding =

new Tensor(

"user.follow2vec.consumer\_avg\_fol\_emb\_200",

DataType.FLOAT

)

private val allFeatures: Seq[Feature[\_]] = Seq(

userId,

userSimclusters,

userF2vConsumerEmbedding

)

val featureContext = new FeatureContext(allFeatures: \_\*)

}