package com.twitter.simclusters\_v2.tweet\_similarity

import com.twitter.ml.api.Feature.{Binary, Continuous, Discrete, SparseContinuous}

import com.twitter.ml.api.util.FDsl.\_

import com.twitter.ml.api.{DataRecord, FeatureContext, IRecordOneToOneAdapter}

import com.twitter.ml.featurestore.catalog.features.recommendations.ProducerSimClustersEmbedding

import com.twitter.ml.featurestore.lib.UserId

import com.twitter.ml.featurestore.lib.data.{PredictionRecord, PredictionRecordAdapter}

import com.twitter.ml.featurestore.lib.entity.Entity

import com.twitter.ml.featurestore.lib.feature.BoundFeatureSet

object TweetSimilarityFeatures {

val QueryTweetId = new Discrete("query\_tweet.id")

val CandidateTweetId = new Discrete("candidate\_tweet.id")

val QueryTweetEmbedding = new SparseContinuous("query\_tweet.simclusters\_embedding")

val CandidateTweetEmbedding = new SparseContinuous("candidate\_tweet.simclusters\_embedding")

val QueryTweetEmbeddingNorm = new Continuous("query\_tweet.embedding\_norm")

val CandidateTweetEmbeddingNorm = new Continuous("candidate\_tweet.embedding\_norm")

val QueryTweetTimestamp = new Discrete("query\_tweet.timestamp")

val CandidateTweetTimestamp = new Discrete("candidate\_tweet.timestamp")

val TweetPairCount = new Discrete("popularity\_count.tweet\_pair")

val QueryTweetCount = new Discrete("popularity\_count.query\_tweet")

val CosineSimilarity = new Continuous("meta.cosine\_similarity")

val Label = new Binary("co-engagement.label")

val FeatureContext: FeatureContext = new FeatureContext(

QueryTweetId,

CandidateTweetId,

QueryTweetEmbedding,

CandidateTweetEmbedding,

QueryTweetEmbeddingNorm,

CandidateTweetEmbeddingNorm,

QueryTweetTimestamp,

CandidateTweetTimestamp,

TweetPairCount,

QueryTweetCount,

CosineSimilarity,

Label

)

def isCoengaged(dataRecord: DataRecord): Boolean = {

dataRecord.getFeatureValue(Label)

}

}

class TweetSimilarityFeaturesStoreConfig(identifier: String) {

val bindingIdentifier: Entity[UserId] = Entity[UserId](identifier)

val featureStoreBoundFeatureSet: BoundFeatureSet = BoundFeatureSet(

ProducerSimClustersEmbedding.FavBasedEmbedding20m145kUpdated.bind(bindingIdentifier))

val predictionRecordAdapter: IRecordOneToOneAdapter[PredictionRecord] =

PredictionRecordAdapter.oneToOne(featureStoreBoundFeatureSet)

}