package com.twitter.timelines.data\_processing.ad\_hoc.earlybird\_ranking.common

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.Feature

import com.twitter.ml.api.FeatureContext

import com.twitter.ml.api.ITransform

import com.twitter.ml.api.transform.CascadeTransform

import com.twitter.ml.api.util.SRichDataRecord

import com.twitter.search.common.features.SearchResultFeature

import com.twitter.search.common.features.TweetFeature

import com.twitter.timelines.prediction.features.itl.ITLFeatures.\_

import scala.collection.JavaConverters.\_

class EarlybirdTrainingRectweetConfiguration extends EarlybirdTrainingConfiguration {

override val labels: Map[String, Feature.Binary] = Map(

"detail\_expanded" -> IS\_CLICKED,

"favorited" -> IS\_FAVORITED,

"open\_linked" -> IS\_OPEN\_LINKED,

"photo\_expanded" -> IS\_PHOTO\_EXPANDED,

"profile\_clicked" -> IS\_PROFILE\_CLICKED,

"replied" -> IS\_REPLIED,

"retweeted" -> IS\_RETWEETED,

"video\_playback50" -> IS\_VIDEO\_PLAYBACK\_50

)

override val PositiveSamplingRate: Double = 0.5

override def featureToSearchResultFeatureMap: Map[Feature[\_], SearchResultFeature] =

super.featureToSearchResultFeatureMap ++ Map(

TEXT\_SCORE -> TweetFeature.TEXT\_SCORE,

REPLY\_COUNT -> TweetFeature.REPLY\_COUNT,

RETWEET\_COUNT -> TweetFeature.RETWEET\_COUNT,

FAV\_COUNT -> TweetFeature.FAVORITE\_COUNT,

HAS\_CARD -> TweetFeature.HAS\_CARD\_FLAG,

HAS\_CONSUMER\_VIDEO -> TweetFeature.HAS\_CONSUMER\_VIDEO\_FLAG,

HAS\_PRO\_VIDEO -> TweetFeature.HAS\_PRO\_VIDEO\_FLAG,

HAS\_VINE -> TweetFeature.HAS\_VINE\_FLAG,

HAS\_PERISCOPE -> TweetFeature.HAS\_PERISCOPE\_FLAG,

HAS\_NATIVE\_IMAGE -> TweetFeature.HAS\_NATIVE\_IMAGE\_FLAG,

HAS\_IMAGE -> TweetFeature.HAS\_IMAGE\_URL\_FLAG,

HAS\_NEWS -> TweetFeature.HAS\_NEWS\_URL\_FLAG,

HAS\_VIDEO -> TweetFeature.HAS\_VIDEO\_URL\_FLAG,

// some features that exist for recap are not available in rectweet

// HAS\_TREND

// HAS\_MULTIPLE\_HASHTAGS\_OR\_TRENDS

// IS\_OFFENSIVE

// IS\_REPLY

// IS\_RETWEET

IS\_AUTHOR\_BOT -> TweetFeature.IS\_USER\_BOT\_FLAG,

IS\_AUTHOR\_SPAM -> TweetFeature.IS\_USER\_SPAM\_FLAG,

IS\_AUTHOR\_NSFW -> TweetFeature.IS\_USER\_NSFW\_FLAG,

// FROM\_VERIFIED\_ACCOUNT

USER\_REP -> TweetFeature.USER\_REPUTATION,

// EMBEDS\_IMPRESSION\_COUNT

// EMBEDS\_URL\_COUNT

// VIDEO\_VIEW\_COUNT

FAV\_COUNT\_V2 -> TweetFeature.FAVORITE\_COUNT\_V2,

RETWEET\_COUNT\_V2 -> TweetFeature.RETWEET\_COUNT\_V2,

REPLY\_COUNT\_V2 -> TweetFeature.REPLY\_COUNT\_V2,

IS\_SENSITIVE -> TweetFeature.IS\_SENSITIVE\_CONTENT,

HAS\_MULTIPLE\_MEDIA -> TweetFeature.HAS\_MULTIPLE\_MEDIA\_FLAG,

IS\_AUTHOR\_PROFILE\_EGG -> TweetFeature.PROFILE\_IS\_EGG\_FLAG,

IS\_AUTHOR\_NEW -> TweetFeature.IS\_USER\_NEW\_FLAG,

NUM\_MENTIONS -> TweetFeature.NUM\_MENTIONS,

NUM\_HASHTAGS -> TweetFeature.NUM\_HASHTAGS,

HAS\_VISIBLE\_LINK -> TweetFeature.HAS\_VISIBLE\_LINK\_FLAG,

HAS\_LINK -> TweetFeature.HAS\_LINK\_FLAG

)

override def derivedFeaturesAdder: CascadeTransform = {

// only LINK\_LANGUAGE availabe in rectweet. no LANGUAGE feature

val linkLanguageTransform = new ITransform {

private val linkLanguageFeature = new Feature.Continuous(TweetFeature.LINK\_LANGUAGE.getName)

override def transformContext(featureContext: FeatureContext): FeatureContext =

featureContext.addFeatures(

linkLanguageFeature

)

override def transform(record: DataRecord): Unit = {

val srecord = SRichDataRecord(record)

srecord.getFeatureValueOpt(LINK\_LANGUAGE).map { link\_language =>

srecord.setFeatureValue(

linkLanguageFeature,

link\_language.toDouble

)

}

}

}

new CascadeTransform(

List(

super.derivedFeaturesAdder,

linkLanguageTransform

).asJava

)

}

}