package com.twitter.timelineranker.recap.model

import com.twitter.escherbird.thriftscala.TweetEntityAnnotation

import com.twitter.timelines.content\_features.v1.thriftscala.{ContentFeatures => ContentFeaturesV1}

import com.twitter.timelines.content\_features.{thriftscala => thrift}

import com.twitter.tweetypie.thriftscala.ConversationControl

import com.twitter.tweetypie.thriftscala.MediaEntity

import com.twitter.tweetypie.thriftscala.SelfThreadMetadata

import scala.util.Failure

import scala.util.Success

import scala.util.{Try => ScalaTry}

case class ContentFeatures(

length: Short,

hasQuestion: Boolean,

numCaps: Short,

numWhiteSpaces: Short,

numNewlines: Option[Short],

videoDurationMs: Option[Int],

bitRate: Option[Int],

aspectRatioNum: Option[Short],

aspectRatioDen: Option[Short],

widths: Option[Seq[Short]],

heights: Option[Seq[Short]],

resizeMethods: Option[Seq[Short]],

numMediaTags: Option[Short],

mediaTagScreenNames: Option[Seq[String]],

emojiTokens: Option[Set[String]],

emoticonTokens: Option[Set[String]],

phrases: Option[Set[String]],

faceAreas: Option[Seq[Int]],

dominantColorRed: Option[Short],

dominantColorBlue: Option[Short],

dominantColorGreen: Option[Short],

numColors: Option[Short],

stickerIds: Option[Seq[Long]],

mediaOriginProviders: Option[Seq[String]],

isManaged: Option[Boolean],

is360: Option[Boolean],

viewCount: Option[Long],

isMonetizable: Option[Boolean],

isEmbeddable: Option[Boolean],

hasSelectedPreviewImage: Option[Boolean],

hasTitle: Option[Boolean],

hasDescription: Option[Boolean],

hasVisitSiteCallToAction: Option[Boolean],

hasAppInstallCallToAction: Option[Boolean],

hasWatchNowCallToAction: Option[Boolean],

media: Option[Seq[MediaEntity]],

dominantColorPercentage: Option[Double],

posUnigrams: Option[Set[String]],

posBigrams: Option[Set[String]],

semanticCoreAnnotations: Option[Seq[TweetEntityAnnotation]],

selfThreadMetadata: Option[SelfThreadMetadata],

tokens: Option[Seq[String]],

tweetText: Option[String],

conversationControl: Option[ConversationControl]) {

def toThrift: thrift.ContentFeatures =

thrift.ContentFeatures.V1(toThriftV1)

def toThriftV1: ContentFeaturesV1 =

ContentFeaturesV1(

length = length,

hasQuestion = hasQuestion,

numCaps = numCaps,

numWhiteSpaces = numWhiteSpaces,

numNewlines = numNewlines,

videoDurationMs = videoDurationMs,

bitRate = bitRate,

aspectRatioNum = aspectRatioNum,

aspectRatioDen = aspectRatioDen,

widths = widths,

heights = heights,

resizeMethods = resizeMethods,

numMediaTags = numMediaTags,

mediaTagScreenNames = mediaTagScreenNames,

emojiTokens = emojiTokens,

emoticonTokens = emoticonTokens,

phrases = phrases,

faceAreas = faceAreas,

dominantColorRed = dominantColorRed,

dominantColorBlue = dominantColorBlue,

dominantColorGreen = dominantColorGreen,

numColors = numColors,

stickerIds = stickerIds,

mediaOriginProviders = mediaOriginProviders,

isManaged = isManaged,

is360 = is360,

viewCount = viewCount,

isMonetizable = isMonetizable,

isEmbeddable = isEmbeddable,

hasSelectedPreviewImage = hasSelectedPreviewImage,

hasTitle = hasTitle,

hasDescription = hasDescription,

hasVisitSiteCallToAction = hasVisitSiteCallToAction,

hasAppInstallCallToAction = hasAppInstallCallToAction,

hasWatchNowCallToAction = hasWatchNowCallToAction,

dominantColorPercentage = dominantColorPercentage,

posUnigrams = posUnigrams,

posBigrams = posBigrams,

semanticCoreAnnotations = semanticCoreAnnotations,

selfThreadMetadata = selfThreadMetadata,

tokens = tokens,

tweetText = tweetText,

conversationControl = conversationControl,

media = media

)

}

object ContentFeatures {

val Empty: ContentFeatures = ContentFeatures(

0.toShort,

false,

0.toShort,

0.toShort,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None,

None

)

def fromThrift(contentFeatures: thrift.ContentFeatures): Option[ContentFeatures] =

contentFeatures match {

case thrift.ContentFeatures.V1(contentFeaturesV1) =>

Some(fromThriftV1(contentFeaturesV1))

case \_ =>

None

}

private val failure =

Failure[ContentFeatures](new Exception("Failure to convert content features from thrift"))

def tryFromThrift(contentFeaturesThrift: thrift.ContentFeatures): ScalaTry[ContentFeatures] =

fromThrift(contentFeaturesThrift) match {

case Some(contentFeatures) => Success[ContentFeatures](contentFeatures)

case None => failure

}

def fromThriftV1(contentFeaturesV1: ContentFeaturesV1): ContentFeatures =

ContentFeatures(

length = contentFeaturesV1.length,

hasQuestion = contentFeaturesV1.hasQuestion,

numCaps = contentFeaturesV1.numCaps,

numWhiteSpaces = contentFeaturesV1.numWhiteSpaces,

numNewlines = contentFeaturesV1.numNewlines,

videoDurationMs = contentFeaturesV1.videoDurationMs,

bitRate = contentFeaturesV1.bitRate,

aspectRatioNum = contentFeaturesV1.aspectRatioNum,

aspectRatioDen = contentFeaturesV1.aspectRatioDen,

widths = contentFeaturesV1.widths,

heights = contentFeaturesV1.heights,

resizeMethods = contentFeaturesV1.resizeMethods,

numMediaTags = contentFeaturesV1.numMediaTags,

mediaTagScreenNames = contentFeaturesV1.mediaTagScreenNames,

emojiTokens = contentFeaturesV1.emojiTokens.map(\_.toSet),

emoticonTokens = contentFeaturesV1.emoticonTokens.map(\_.toSet),

phrases = contentFeaturesV1.phrases.map(\_.toSet),

faceAreas = contentFeaturesV1.faceAreas,

dominantColorRed = contentFeaturesV1.dominantColorRed,

dominantColorBlue = contentFeaturesV1.dominantColorBlue,

dominantColorGreen = contentFeaturesV1.dominantColorGreen,

numColors = contentFeaturesV1.numColors,

stickerIds = contentFeaturesV1.stickerIds,

mediaOriginProviders = contentFeaturesV1.mediaOriginProviders,

isManaged = contentFeaturesV1.isManaged,

is360 = contentFeaturesV1.is360,

viewCount = contentFeaturesV1.viewCount,

isMonetizable = contentFeaturesV1.isMonetizable,

isEmbeddable = contentFeaturesV1.isEmbeddable,

hasSelectedPreviewImage = contentFeaturesV1.hasSelectedPreviewImage,

hasTitle = contentFeaturesV1.hasTitle,

hasDescription = contentFeaturesV1.hasDescription,

hasVisitSiteCallToAction = contentFeaturesV1.hasVisitSiteCallToAction,

hasAppInstallCallToAction = contentFeaturesV1.hasAppInstallCallToAction,

hasWatchNowCallToAction = contentFeaturesV1.hasWatchNowCallToAction,

dominantColorPercentage = contentFeaturesV1.dominantColorPercentage,

posUnigrams = contentFeaturesV1.posUnigrams.map(\_.toSet),

posBigrams = contentFeaturesV1.posBigrams.map(\_.toSet),

semanticCoreAnnotations = contentFeaturesV1.semanticCoreAnnotations,

selfThreadMetadata = contentFeaturesV1.selfThreadMetadata,

tokens = contentFeaturesV1.tokens.map(\_.toSeq),

tweetText = contentFeaturesV1.tweetText,

conversationControl = contentFeaturesV1.conversationControl,

media = contentFeaturesV1.media

)

}