package com.twitter.visibility.builder.tweets

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.TrustedFriendsListId

import com.twitter.visibility.common.TrustedFriendsSource

import com.twitter.visibility.features.TweetIsTrustedFriendTweet

import com.twitter.visibility.features.ViewerIsTrustedFriendOfTweetAuthor

import com.twitter.visibility.features.ViewerIsTrustedFriendTweetAuthor

class TrustedFriendsFeatures(trustedFriendsSource: TrustedFriendsSource) {

private[builder] def viewerIsTrustedFriend(

tweet: Tweet,

viewerId: Option[Long]

): Stitch[Boolean] =

(trustedFriendsListId(tweet), viewerId) match {

case (Some(tfListId), Some(userId)) =>

trustedFriendsSource.isTrustedFriend(tfListId, userId)

case \_ => Stitch.False

}

private[builder] def viewerIsTrustedFriendListOwner(

tweet: Tweet,

viewerId: Option[Long]

): Stitch[Boolean] =

(trustedFriendsListId(tweet), viewerId) match {

case (Some(tfListId), Some(userId)) =>

trustedFriendsSource.isTrustedFriendListOwner(tfListId, userId)

case \_ => Stitch.False

}

private[builder] def trustedFriendsListId(tweet: Tweet): Option[TrustedFriendsListId] =

tweet.trustedFriendsControl.map(\_.trustedFriendsListId)

def forTweet(

tweet: Tweet,

viewerId: Option[Long]

): FeatureMapBuilder => FeatureMapBuilder = {

\_.withConstantFeature(

TweetIsTrustedFriendTweet,

tweet.trustedFriendsControl.isDefined

).withFeature(

ViewerIsTrustedFriendTweetAuthor,

viewerIsTrustedFriendListOwner(tweet, viewerId)

).withFeature(

ViewerIsTrustedFriendOfTweetAuthor,

viewerIsTrustedFriend(tweet, viewerId)

)

}

def forTweetOnly(tweet: Tweet): FeatureMapBuilder => FeatureMapBuilder = {

\_.withConstantFeature(TweetIsTrustedFriendTweet, tweet.trustedFriendsControl.isDefined)

}

}