package com.twitter.frigate.pushservice.predicate

import com.twitter.frigate.common.base.\_

import com.twitter.frigate.data\_pipeline.features\_common.MrRequestContextForFeatureStore

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.ml.featurestore.catalog.entities.core.Tweet

import com.twitter.ml.featurestore.catalog.features.core.Tweet.Text

import com.twitter.ml.featurestore.lib.TweetId

import com.twitter.ml.featurestore.lib.dynamic.DynamicFeatureStoreClient

import com.twitter.ml.featurestore.lib.online.FeatureStoreRequest

import com.twitter.util.Future

object PostRankingPredicateHelper {

val tweetTextFeature = "tweet.core.tweet.text"

def getTweetText(

candidate: PushCandidate with TweetCandidate,

dynamicClient: DynamicFeatureStoreClient[MrRequestContextForFeatureStore]

): Future[String] = {

if (candidate.categoricalFeatures.contains(tweetTextFeature)) {

Future.value(candidate.categoricalFeatures.getOrElse(tweetTextFeature, ""))

} else {

val candidateTweetEntity = Tweet.withId(TweetId(candidate.tweetId))

val featureStoreRequests = Seq(

FeatureStoreRequest(

entityIds = Seq(candidateTweetEntity)

))

val predictionRecords = dynamicClient(

featureStoreRequests,

requestContext = candidate.target.mrRequestContextForFeatureStore)

predictionRecords.map { records =>

val tweetText = records.head

.getFeatureValue(candidateTweetEntity, Text).getOrElse(

""

)

candidate.categoricalFeatures(tweetTextFeature) = tweetText

tweetText

}

}

}

def getTweetWordLength(tweetText: String): Double = {

val tweetTextWithoutUrl: String =

tweetText.replaceAll("https?://\\S+\\s?", "").replaceAll("[\\s]+", " ")

tweetTextWithoutUrl.trim().split(" ").length.toDouble

}

}