package com.twitter.search.earlybird.search.relevance;

import java.util.Arrays;

import java.util.List;

import com.google.common.collect.Lists;

import com.twitter.search.common.constants.SearchCardType;

import com.twitter.search.common.constants.thriftjava.ThriftLanguage;

public class LinearScoringData {

public static final float NO\_BOOST\_VALUE = 1.0f;

// A signal value so we can tell if something is unset, also used in explanation.

public static final int UNSET\_SIGNAL\_VALUE = -999;

//This is somewhat arbitrary, and is here so that we have some limit on

//how many offline experimental features we support per query

public static final int MAX\_OFFLINE\_EXPERIMENTAL\_FIELDS = 5;

public enum SkipReason {

NOT\_SKIPPED,

ANTIGAMING,

LOW\_REPUTATION,

LOW\_TEXT\_SCORE,

LOW\_RETWEET\_COUNT,

LOW\_FAV\_COUNT,

SOCIAL\_FILTER,

LOW\_FINAL\_SCORE

}

// When you add fields here, make sure you also update the clear() function.

public double luceneScore;

public double textScore;

//I am not sure why this has to be double...

public double tokenAt140DividedByNumTokensBucket;

public double userRep;

public double parusScore;

public final double[] offlineExpFeatureValues = new double[MAX\_OFFLINE\_EXPERIMENTAL\_FIELDS];

// v1 engagement counters

public double retweetCountPostLog2;

public double favCountPostLog2;

public double replyCountPostLog2;

public double embedsImpressionCount;

public double embedsUrlCount;

public double videoViewCount;

// v2 engagement counters (that have a v1 counter part)

public double retweetCountV2;

public double favCountV2;

public double replyCountV2;

public double embedsImpressionCountV2;

public double embedsUrlCountV2;

public double videoViewCountV2;

// pure v2 engagement counters, they started v2 only

public double quotedCount;

public double weightedRetweetCount;

public double weightedReplyCount;

public double weightedFavCount;

public double weightedQuoteCount;

// card related properties

public boolean hasCard;

public byte cardType;

public boolean hasUrl;

public boolean isReply;

public boolean isRetweet;

public boolean isOffensive;

public boolean hasTrend;

public boolean isFromVerifiedAccount;

public boolean isFromBlueVerifiedAccount;

public boolean isUserSpam;

public boolean isUserNSFW;

public boolean isUserBot;

public boolean isUserAntiSocial;

public boolean hasVisibleLink;

public double luceneContrib;

public double reputationContrib;

public double textScoreContrib;

public double favContrib;

public double replyContrib;

public double multipleReplyContrib;

public double retweetContrib;

public double parusContrib;

public final double[] offlineExpFeatureContributions =

new double[MAX\_OFFLINE\_EXPERIMENTAL\_FIELDS];

public double embedsImpressionContrib;

public double embedsUrlContrib;

public double videoViewContrib;

public double quotedContrib;

public double hasUrlContrib;

public double isReplyContrib;

public double isFollowRetweetContrib;

public double isTrustedRetweetContrib;

// Value passed in the request (ThriftRankingParams.querySpecificScoreAdjustments)

public double querySpecificScore;

// Value passed in the request (ThriftRankingParams.authorSpecificScoreAdjustments)

public double authorSpecificScore;

public double normalizedLuceneScore;

public int tweetLangId;

public double uiLangMult;

public double userLangMult;

public boolean hasDifferentLang;

public boolean hasEnglishTweetAndDifferentUILang;

public boolean hasEnglishUIAndDifferentTweetLang;

public int tweetAgeInSeconds;

public double ageDecayMult;

// Intermediate scores

public double scoreBeforeBoost;

public double scoreAfterBoost;

public double scoreFinal;

public double scoreReturned;

public SkipReason skipReason;

public boolean isTrusted;

public boolean isFollow;

public boolean spamUserDampApplied;

public boolean nsfwUserDampApplied;

public boolean botUserDampApplied;

public boolean trustedCircleBoostApplied;

public boolean directFollowBoostApplied;

public boolean outOfNetworkReplyPenaltyApplied;

public boolean hasMultipleHashtagsOrTrends;

public boolean tweetHasTrendsBoostApplied;

public boolean tweetFromVerifiedAccountBoostApplied;

public boolean tweetFromBlueVerifiedAccountBoostApplied;

public boolean hasCardBoostApplied;

public boolean cardDomainMatchBoostApplied;

public boolean cardAuthorMatchBoostApplied;

public boolean cardTitleMatchBoostApplied;

public boolean cardDescriptionMatchBoostApplied;

public List<String> hitFields;

public boolean hasNoTextHitDemotionApplied;

public boolean hasUrlOnlyHitDemotionApplied;

public boolean hasNameOnlyHitDemotionApplied;

public boolean hasSeparateTextAndNameHitDemotionApplied;

public boolean hasSeparateTextAndUrlHitDemotionApplied;

public long fromUserId;

// This is actually retweet status ID, or the ID of the original tweet being (natively) retweeted

public long sharedStatusId;

public long referenceAuthorId; // SEARCH-8564

public boolean isSelfTweet;

public boolean selfTweetBoostApplied;

public double selfTweetMult;

public boolean hasImageUrl;

public boolean hasVideoUrl;

public boolean hasMedialUrlBoostApplied;

public boolean hasNewsUrl;

public boolean hasNewsUrlBoostApplied;

public boolean hasConsumerVideo;

public boolean hasProVideo;

public boolean hasVine;

public boolean hasPeriscope;

public boolean hasNativeImage;

public boolean isNullcast;

public boolean hasQuote;

public boolean isSensitiveContent;

public boolean hasMultipleMediaFlag;

public boolean profileIsEggFlag;

public boolean isUserNewFlag;

public int numMentions;

public int numHashtags;

public int linkLanguage;

public int prevUserTweetEngagement;

public boolean isComposerSourceCamera;

// health model scores by HML

public double toxicityScore; // go/toxicity

public double pBlockScore; // go/pblock

public double pSpammyTweetScore; // go/pspammytweet

public double pReportedTweetScore; // go/preportedtweet

public double spammyTweetContentScore; // go/spammy-tweet-content

public double experimentalHealthModelScore1;

public double experimentalHealthModelScore2;

public double experimentalHealthModelScore3;

public double experimentalHealthModelScore4;

public LinearScoringData() {

hitFields = Lists.newArrayList();

clear();

}

// the following three counters were added later and they got denormalized in standard way,

// you can choose to apply scalding (for legacy LinearScoringFunction) or

// not apply (for returning in metadata and display in debug).

public double getEmbedsImpressionCount(boolean scaleForScoring) {

return scaleForScoring ? logWith0(embedsImpressionCount) : embedsImpressionCount;

}

public double getEmbedsUrlCount(boolean scaleForScoring) {

return scaleForScoring ? logWith0(embedsUrlCount) : embedsUrlCount;

}

public double getVideoViewCount(boolean scaleForScoring) {

return scaleForScoring ? logWith0(videoViewCount) : videoViewCount;

}

private static double logWith0(double value) {

return value > 0 ? Math.log(value) : 0.0;

}

/\*\*

\* Returns a string description of all data stored in this instance.

\*/

public String getPropertyExplanation() {

StringBuilder sb = new StringBuilder();

sb.append(hasCard ? "CARD " + SearchCardType.cardTypeFromByteValue(cardType) : "");

sb.append(hasUrl ? "URL " : "");

sb.append(isReply ? "REPLY " : "");

sb.append(isRetweet ? "RETWEET " : "");

sb.append(isOffensive ? "OFFENSIVE " : "");

sb.append(hasTrend ? "TREND " : "");

sb.append(hasMultipleHashtagsOrTrends ? "HASHTAG/TREND+ " : "");

sb.append(isFromVerifiedAccount ? "VERIFIED " : "");

sb.append(isFromBlueVerifiedAccount ? "BLUE\_VERIFIED " : "");

sb.append(isUserSpam ? "SPAM " : "");

sb.append(isUserNSFW ? "NSFW " : "");

sb.append(isUserBot ? "BOT " : "");

sb.append(isUserAntiSocial ? "ANTISOCIAL " : "");

sb.append(isTrusted ? "TRUSTED " : "");

sb.append(isFollow ? "FOLLOW " : "");

sb.append(isSelfTweet ? "SELF " : "");

sb.append(hasImageUrl ? "IMAGE " : "");

sb.append(hasVideoUrl ? "VIDEO " : "");

sb.append(hasNewsUrl ? "NEWS " : "");

sb.append(isNullcast ? "NULLCAST" : "");

sb.append(hasQuote ? "QUOTE" : "");

sb.append(isComposerSourceCamera ? "Composer Source: CAMERA" : "");

sb.append(favCountPostLog2 > 0 ? "Faves:" + favCountPostLog2 + " " : "");

sb.append(retweetCountPostLog2 > 0 ? "Retweets:" + retweetCountPostLog2 + " " : "");

sb.append(replyCountPostLog2 > 0 ? "Replies:" + replyCountPostLog2 + " " : "");

sb.append(getEmbedsImpressionCount(false) > 0

? "Embedded Imps:" + getEmbedsImpressionCount(false) + " " : "");

sb.append(getEmbedsUrlCount(false) > 0

? "Embedded Urls:" + getEmbedsUrlCount(false) + " " : "");

sb.append(getVideoViewCount(false) > 0

? "Video views:" + getVideoViewCount(false) + " " : "");

sb.append(weightedRetweetCount > 0 ? "Weighted Retweets:"

+ ((int) weightedRetweetCount) + " " : "");

sb.append(weightedReplyCount > 0

? "Weighted Replies:" + ((int) weightedReplyCount) + " " : "");

sb.append(weightedFavCount > 0

? "Weighted Faves:" + ((int) weightedFavCount) + " " : "");

sb.append(weightedQuoteCount > 0

? "Weighted Quotes:" + ((int) weightedQuoteCount) + " " : "");

return sb.toString();

}

/\*\*

\* Resets all data stored in this instance.

\*/

public void clear() {

luceneScore = UNSET\_SIGNAL\_VALUE;

textScore = UNSET\_SIGNAL\_VALUE;

tokenAt140DividedByNumTokensBucket = UNSET\_SIGNAL\_VALUE;

userRep = UNSET\_SIGNAL\_VALUE;

retweetCountPostLog2 = UNSET\_SIGNAL\_VALUE;

favCountPostLog2 = UNSET\_SIGNAL\_VALUE;

replyCountPostLog2 = UNSET\_SIGNAL\_VALUE;

parusScore = UNSET\_SIGNAL\_VALUE;

Arrays.fill(offlineExpFeatureValues, 0);

embedsImpressionCount = UNSET\_SIGNAL\_VALUE;

embedsUrlCount = UNSET\_SIGNAL\_VALUE;

videoViewCount = UNSET\_SIGNAL\_VALUE;

// v2 engagement, these each have a v1 counterpart

retweetCountV2 = UNSET\_SIGNAL\_VALUE;

favCountV2 = UNSET\_SIGNAL\_VALUE;

replyCountV2 = UNSET\_SIGNAL\_VALUE;

embedsImpressionCountV2 = UNSET\_SIGNAL\_VALUE;

embedsUrlCountV2 = UNSET\_SIGNAL\_VALUE;

videoViewCountV2 = UNSET\_SIGNAL\_VALUE;

// new engagement counters, they only have one version with the v2 normalizer

quotedCount = UNSET\_SIGNAL\_VALUE;

weightedRetweetCount = UNSET\_SIGNAL\_VALUE;

weightedReplyCount = UNSET\_SIGNAL\_VALUE;

weightedFavCount = UNSET\_SIGNAL\_VALUE;

weightedQuoteCount = UNSET\_SIGNAL\_VALUE;

hasUrl = false;

isReply = false;

isRetweet = false;

isOffensive = false;

hasTrend = false;

isFromVerifiedAccount = false;

isFromBlueVerifiedAccount = false;

isUserSpam = false;

isUserNSFW = false;

isUserBot = false;

isUserAntiSocial = false;

hasVisibleLink = false;

isNullcast = false;

luceneContrib = UNSET\_SIGNAL\_VALUE;

reputationContrib = UNSET\_SIGNAL\_VALUE;

textScoreContrib = UNSET\_SIGNAL\_VALUE;

replyContrib = UNSET\_SIGNAL\_VALUE;

multipleReplyContrib = UNSET\_SIGNAL\_VALUE;

retweetContrib = UNSET\_SIGNAL\_VALUE;

favContrib = UNSET\_SIGNAL\_VALUE;

parusContrib = UNSET\_SIGNAL\_VALUE;

Arrays.fill(offlineExpFeatureContributions, 0);

embedsImpressionContrib = UNSET\_SIGNAL\_VALUE;

embedsUrlContrib = UNSET\_SIGNAL\_VALUE;

videoViewContrib = UNSET\_SIGNAL\_VALUE;

hasUrlContrib = UNSET\_SIGNAL\_VALUE;

isReplyContrib = UNSET\_SIGNAL\_VALUE;

querySpecificScore = UNSET\_SIGNAL\_VALUE;

authorSpecificScore = UNSET\_SIGNAL\_VALUE;

normalizedLuceneScore = NO\_BOOST\_VALUE;

tweetLangId = ThriftLanguage.UNKNOWN.getValue();

uiLangMult = NO\_BOOST\_VALUE;

userLangMult = NO\_BOOST\_VALUE;

hasDifferentLang = false;

hasEnglishTweetAndDifferentUILang = false;

hasEnglishUIAndDifferentTweetLang = false;

tweetAgeInSeconds = 0;

ageDecayMult = NO\_BOOST\_VALUE;

// Intermediate scores

scoreBeforeBoost = UNSET\_SIGNAL\_VALUE;

scoreAfterBoost = UNSET\_SIGNAL\_VALUE;

scoreFinal = UNSET\_SIGNAL\_VALUE;

scoreReturned = UNSET\_SIGNAL\_VALUE;

skipReason = SkipReason.NOT\_SKIPPED;

isTrusted = false; // Set later

isFollow = false; // Set later

trustedCircleBoostApplied = false;

directFollowBoostApplied = false;

outOfNetworkReplyPenaltyApplied = false;

hasMultipleHashtagsOrTrends = false;

spamUserDampApplied = false;

nsfwUserDampApplied = false;

botUserDampApplied = false;

tweetHasTrendsBoostApplied = false;

tweetFromVerifiedAccountBoostApplied = false;

tweetFromBlueVerifiedAccountBoostApplied = false;

fromUserId = UNSET\_SIGNAL\_VALUE;

sharedStatusId = UNSET\_SIGNAL\_VALUE;

referenceAuthorId = UNSET\_SIGNAL\_VALUE;

isSelfTweet = false;

selfTweetBoostApplied = false;

selfTweetMult = NO\_BOOST\_VALUE;

trustedCircleBoostApplied = false;

directFollowBoostApplied = false;

hasImageUrl = false;

hasVideoUrl = false;

hasMedialUrlBoostApplied = false;

hasNewsUrl = false;

hasNewsUrlBoostApplied = false;

hasCard = false;

cardType = SearchCardType.UNKNOWN.getByteValue();

hasCardBoostApplied = false;

cardDomainMatchBoostApplied = false;

cardAuthorMatchBoostApplied = false;

cardTitleMatchBoostApplied = false;

cardDescriptionMatchBoostApplied = false;

hitFields.clear();

hasNoTextHitDemotionApplied = false;

hasUrlOnlyHitDemotionApplied = false;

hasNameOnlyHitDemotionApplied = false;

hasSeparateTextAndNameHitDemotionApplied = false;

hasSeparateTextAndUrlHitDemotionApplied = false;

hasConsumerVideo = false;

hasProVideo = false;

hasVine = false;

hasPeriscope = false;

hasNativeImage = false;

isSensitiveContent = false;

hasMultipleMediaFlag = false;

profileIsEggFlag = false;

numMentions = 0;

numHashtags = 0;

isUserNewFlag = false;

linkLanguage = 0;

prevUserTweetEngagement = 0;

isComposerSourceCamera = false;

// health model scores by HML

toxicityScore = UNSET\_SIGNAL\_VALUE;

pBlockScore = UNSET\_SIGNAL\_VALUE;

pSpammyTweetScore = UNSET\_SIGNAL\_VALUE;

pReportedTweetScore = UNSET\_SIGNAL\_VALUE;

spammyTweetContentScore = UNSET\_SIGNAL\_VALUE;

experimentalHealthModelScore1 = UNSET\_SIGNAL\_VALUE;

experimentalHealthModelScore2 = UNSET\_SIGNAL\_VALUE;

experimentalHealthModelScore3 = UNSET\_SIGNAL\_VALUE;

experimentalHealthModelScore4 = UNSET\_SIGNAL\_VALUE;

}

}