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

import java.util.Arrays;

import java.util.Map;

import com.google.common.annotations.VisibleForTesting;

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

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

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.ranking.thriftjava.ThriftAgeDecayRankingParams;

import com.twitter.search.common.ranking.thriftjava.ThriftCardRankingParams;

import com.twitter.search.common.ranking.thriftjava.ThriftRankingParams;

import com.twitter.search.common.util.lang.ThriftLanguageUtil;

import com.twitter.search.earlybird.thrift.ThriftSearchQuery;

import com.twitter.search.earlybird.thrift.ThriftSocialFilterType;

/\*

\* The class for all query specific parameters, including the parameters from the relevanceOptions and

\* values that are extracted from the request itself.

\*/

public class LinearScoringParams {

public static final double DEFAULT\_FEATURE\_WEIGHT = 0;

public static final double DEFAULT\_FEATURE\_MIN\_VAL = 0;

public static final double DEFAULT\_NO\_BOOST = 1.0;

@VisibleForTesting

static final SearchCounter NULL\_USER\_LANGS\_KEY =

SearchCounter.export("linear\_scoring\_params\_null\_user\_langs\_key");

public final double luceneWeight;

public final double textScoreWeight;

public final double textScoreMinVal;

public final double retweetWeight;

public final double retweetMinVal;

public final double favWeight;

public final double favMinVal;

public final double replyWeight;

public final double multipleReplyWeight;

public final double multipleReplyMinVal;

public final double isReplyWeight;

public final double parusWeight;

public final double embedsImpressionWeight;

public final double embedsUrlWeight;

public final double videoViewWeight;

public final double quotedCountWeight;

public final double[] rankingOfflineExpWeights =

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

public final boolean applyBoosts;

// Storing ranking params for cards, avoid using maps for faster lookup

public final double[] hasCardBoosts = new double[SearchCardType.values().length];

public final double[] cardDomainMatchBoosts = new double[SearchCardType.values().length];

public final double[] cardAuthorMatchBoosts = new double[SearchCardType.values().length];

public final double[] cardTitleMatchBoosts = new double[SearchCardType.values().length];

public final double[] cardDescriptionMatchBoosts = new double[SearchCardType.values().length];

public final double urlWeight;

public final double reputationWeight;

public final double reputationMinVal;

public final double followRetweetWeight;

public final double trustedRetweetWeight;

// Adjustments for specific tweets (tweetId -> score)

public final Map<Long, Double> querySpecificScoreAdjustments;

// Adjustments for tweets posted by specific authors (userId -> score)

public final Map<Long, Double> authorSpecificScoreAdjustments;

public final double offensiveDamping;

public final double spamUserDamping;

public final double nsfwUserDamping;

public final double botUserDamping;

public final double trustedCircleBoost;

public final double directFollowBoost;

public final double minScore;

public final boolean applyFiltersAlways;

public final boolean useLuceneScoreAsBoost;

public final double maxLuceneScoreBoost;

public final double langEnglishTweetDemote;

public final double langEnglishUIDemote;

public final double langDefaultDemote;

public final boolean useUserLanguageInfo;

public final double unknownLanguageBoost;

public final double outOfNetworkReplyPenalty;

public final boolean useAgeDecay;

public final double ageDecayHalflife;

public final double ageDecayBase;

public final double ageDecaySlope;

// hit attribute demotions

public final boolean enableHitDemotion;

public final double noTextHitDemotion;

public final double urlOnlyHitDemotion;

public final double nameOnlyHitDemotion;

public final double separateTextAndNameHitDemotion;

public final double separateTextAndUrlHitDemotion;

// trends related params

public final double tweetHasTrendBoost;

public final double multipleHashtagsOrTrendsDamping;

public final double tweetFromVerifiedAccountBoost;

public final double tweetFromBlueVerifiedAccountBoost;

public final ThriftSocialFilterType socialFilterType;

public final int uiLangId;

// Confidences of the understandability of different languages for this user.

public final double[] userLangs = new double[ThriftLanguage.values().length];

public final long searcherId;

public final double selfTweetBoost;

public final double tweetHasMediaUrlBoost;

public final double tweetHasNewsUrlBoost;

// whether we need meta-data for replies what the reply is to.

public final boolean getInReplyToStatusId;

// Initialize from a ranking parameter

public LinearScoringParams(ThriftSearchQuery searchQuery, ThriftRankingParams params) {

// weights

luceneWeight = params.isSetLuceneScoreParams()

? params.getLuceneScoreParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

textScoreWeight = params.isSetTextScoreParams()

? params.getTextScoreParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

retweetWeight = params.isSetRetweetCountParams()

? params.getRetweetCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

favWeight = params.isSetFavCountParams()

? params.getFavCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

replyWeight = params.isSetReplyCountParams()

? params.getReplyCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

multipleReplyWeight = params.isSetMultipleReplyCountParams()

? params.getMultipleReplyCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

parusWeight = params.isSetParusScoreParams()

? params.getParusScoreParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

for (int i = 0; i < LinearScoringData.MAX\_OFFLINE\_EXPERIMENTAL\_FIELDS; i++) {

Byte featureTypeByte = (byte) i;

// default weight is 0, thus contribution for unset feature value will be 0.

rankingOfflineExpWeights[i] = params.getOfflineExperimentalFeatureRankingParamsSize() > 0

&& params.getOfflineExperimentalFeatureRankingParams().containsKey(featureTypeByte)

? params.getOfflineExperimentalFeatureRankingParams().get(featureTypeByte).getWeight()

: DEFAULT\_FEATURE\_WEIGHT;

}

embedsImpressionWeight = params.isSetEmbedsImpressionCountParams()

? params.getEmbedsImpressionCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

embedsUrlWeight = params.isSetEmbedsUrlCountParams()

? params.getEmbedsUrlCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

videoViewWeight = params.isSetVideoViewCountParams()

? params.getVideoViewCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

quotedCountWeight = params.isSetQuotedCountParams()

? params.getQuotedCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

applyBoosts = params.isApplyBoosts();

// configure card values

Arrays.fill(hasCardBoosts, DEFAULT\_NO\_BOOST);

Arrays.fill(cardAuthorMatchBoosts, DEFAULT\_NO\_BOOST);

Arrays.fill(cardDomainMatchBoosts, DEFAULT\_NO\_BOOST);

Arrays.fill(cardTitleMatchBoosts, DEFAULT\_NO\_BOOST);

Arrays.fill(cardDescriptionMatchBoosts, DEFAULT\_NO\_BOOST);

if (params.isSetCardRankingParams()) {

for (SearchCardType cardType : SearchCardType.values()) {

byte cardTypeIndex = cardType.getByteValue();

ThriftCardRankingParams rankingParams = params.getCardRankingParams().get(cardTypeIndex);

if (rankingParams != null) {

hasCardBoosts[cardTypeIndex] = rankingParams.getHasCardBoost();

cardAuthorMatchBoosts[cardTypeIndex] = rankingParams.getAuthorMatchBoost();

cardDomainMatchBoosts[cardTypeIndex] = rankingParams.getDomainMatchBoost();

cardTitleMatchBoosts[cardTypeIndex] = rankingParams.getTitleMatchBoost();

cardDescriptionMatchBoosts[cardTypeIndex] = rankingParams.getDescriptionMatchBoost();

}

}

}

urlWeight = params.isSetUrlParams()

? params.getUrlParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

reputationWeight = params.isSetReputationParams()

? params.getReputationParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

isReplyWeight = params.isSetIsReplyParams()

? params.getIsReplyParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

followRetweetWeight = params.isSetDirectFollowRetweetCountParams()

? params.getDirectFollowRetweetCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

trustedRetweetWeight = params.isSetTrustedCircleRetweetCountParams()

? params.getTrustedCircleRetweetCountParams().getWeight() : DEFAULT\_FEATURE\_WEIGHT;

querySpecificScoreAdjustments = params.getQuerySpecificScoreAdjustments();

authorSpecificScoreAdjustments = params.getAuthorSpecificScoreAdjustments();

// min/max filters

textScoreMinVal = params.isSetTextScoreParams()

? params.getTextScoreParams().getMin() : DEFAULT\_FEATURE\_MIN\_VAL;

reputationMinVal = params.isSetReputationParams()

? params.getReputationParams().getMin() : DEFAULT\_FEATURE\_MIN\_VAL;

multipleReplyMinVal = params.isSetMultipleReplyCountParams()

? params.getMultipleReplyCountParams().getMin() : DEFAULT\_FEATURE\_MIN\_VAL;

retweetMinVal = params.isSetRetweetCountParams() && params.getRetweetCountParams().isSetMin()

? params.getRetweetCountParams().getMin() : DEFAULT\_FEATURE\_MIN\_VAL;

favMinVal = params.isSetFavCountParams() && params.getFavCountParams().isSetMin()

? params.getFavCountParams().getMin() : DEFAULT\_FEATURE\_MIN\_VAL;

// boosts

spamUserDamping = params.isSetSpamUserBoost() ? params.getSpamUserBoost() : 1.0;

nsfwUserDamping = params.isSetNsfwUserBoost() ? params.getNsfwUserBoost() : 1.0;

botUserDamping = params.isSetBotUserBoost() ? params.getBotUserBoost() : 1.0;

offensiveDamping = params.getOffensiveBoost();

trustedCircleBoost = params.getInTrustedCircleBoost();

directFollowBoost = params.getInDirectFollowBoost();

// language boosts

langEnglishTweetDemote = params.getLangEnglishTweetBoost();

langEnglishUIDemote = params.getLangEnglishUIBoost();

langDefaultDemote = params.getLangDefaultBoost();

useUserLanguageInfo = params.isUseUserLanguageInfo();

unknownLanguageBoost = params.getUnknownLanguageBoost();

// hit demotions

enableHitDemotion = params.isEnableHitDemotion();

noTextHitDemotion = params.getNoTextHitDemotion();

urlOnlyHitDemotion = params.getUrlOnlyHitDemotion();

nameOnlyHitDemotion = params.getNameOnlyHitDemotion();

separateTextAndNameHitDemotion = params.getSeparateTextAndNameHitDemotion();

separateTextAndUrlHitDemotion = params.getSeparateTextAndUrlHitDemotion();

outOfNetworkReplyPenalty = params.getOutOfNetworkReplyPenalty();

if (params.isSetAgeDecayParams()) {

// new age decay settings

ThriftAgeDecayRankingParams ageDecayParams = params.getAgeDecayParams();

ageDecaySlope = ageDecayParams.getSlope();

ageDecayHalflife = ageDecayParams.getHalflife();

ageDecayBase = ageDecayParams.getBase();

useAgeDecay = true;

} else if (params.isSetDeprecatedAgeDecayBase()

&& params.isSetDeprecatedAgeDecayHalflife()

&& params.isSetDeprecatedAgeDecaySlope()) {

ageDecaySlope = params.getDeprecatedAgeDecaySlope();

ageDecayHalflife = params.getDeprecatedAgeDecayHalflife();

ageDecayBase = params.getDeprecatedAgeDecayBase();

useAgeDecay = true;

} else {

ageDecaySlope = 0.0;

ageDecayHalflife = 0.0;

ageDecayBase = 0.0;

useAgeDecay = false;

}

// trends

tweetHasTrendBoost = params.getTweetHasTrendBoost();

multipleHashtagsOrTrendsDamping = params.getMultipleHashtagsOrTrendsBoost();

// verified accounts

tweetFromVerifiedAccountBoost = params.getTweetFromVerifiedAccountBoost();

tweetFromBlueVerifiedAccountBoost = params.getTweetFromBlueVerifiedAccountBoost();

// score filter

minScore = params.getMinScore();

applyFiltersAlways = params.isApplyFiltersAlways();

useLuceneScoreAsBoost = params.isUseLuceneScoreAsBoost();

maxLuceneScoreBoost = params.getMaxLuceneScoreBoost();

searcherId = searchQuery.isSetSearcherId() ? searchQuery.getSearcherId() : -1;

selfTweetBoost = params.getSelfTweetBoost();

socialFilterType = searchQuery.getSocialFilterType();

// the UI language and the confidences of the languages user can understand.

if (!searchQuery.isSetUiLang() || searchQuery.getUiLang().isEmpty()) {

uiLangId = ThriftLanguage.UNKNOWN.getValue();

} else {

uiLangId = ThriftLanguageUtil.getThriftLanguageOf(searchQuery.getUiLang()).getValue();

}

if (searchQuery.getUserLangsSize() > 0) {

for (Map.Entry<ThriftLanguage, Double> lang : searchQuery.getUserLangs().entrySet()) {

ThriftLanguage thriftLanguage = lang.getKey();

// SEARCH-13441

if (thriftLanguage != null) {

userLangs[thriftLanguage.getValue()] = lang.getValue();

} else {

NULL\_USER\_LANGS\_KEY.increment();

}

}

}

// For now, we will use the same boost for both image, and video.

tweetHasMediaUrlBoost = params.getTweetHasImageUrlBoost();

tweetHasNewsUrlBoost = params.getTweetHasNewsUrlBoost();

getInReplyToStatusId =

searchQuery.isSetResultMetadataOptions()

&& searchQuery.getResultMetadataOptions().isSetGetInReplyToStatusId()

&& searchQuery.getResultMetadataOptions().isGetInReplyToStatusId();

}

}