package com.twitter.home\_mixer.functional\_component.decorator.builder

import com.twitter.conversions.DurationOps.\_

import com.twitter.home\_mixer.model.HomeFeatures.\_

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.BasicTopicContextFunctionalityType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.RecommendationTopicContextFunctionalityType

import com.twitter.timelinemixer.injection.model.candidate.SemanticCoreFeatures

import com.twitter.tweetypie.{thriftscala => tpt}

object HomeTweetTypePredicates {

/\*\*

\* IMPORTANT: Please avoid logging tweet types that are tied to sensitive

\* internal author information / labels (e.g. blink labels, abuse labels, or geo-location).

\*/

private[this] val CandidatePredicates: Seq[(String, FeatureMap => Boolean)] = Seq(

("with\_candidate", \_ => true),

("retweet", \_.getOrElse(IsRetweetFeature, false)),

("reply", \_.getOrElse(InReplyToTweetIdFeature, None).nonEmpty),

("image", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasImage)),

("video", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasVideo)),

("link", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasVisibleLink)),

("quote", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasQuote.contains(true))),

("like\_social\_context", \_.getOrElse(NonSelfFavoritedByUserIdsFeature, Seq.empty).nonEmpty),

("protected", \_.getOrElse(EarlybirdFeature, None).exists(\_.isProtected)),

(

"has\_exclusive\_conversation\_author\_id",

\_.getOrElse(ExclusiveConversationAuthorIdFeature, None).nonEmpty),

("is\_eligible\_for\_connect\_boost", \_ => false),

("hashtag", \_.getOrElse(EarlybirdFeature, None).exists(\_.numHashtags > 0)),

("has\_scheduled\_space", \_.getOrElse(AudioSpaceMetaDataFeature, None).exists(\_.isScheduled)),

("has\_recorded\_space", \_.getOrElse(AudioSpaceMetaDataFeature, None).exists(\_.isRecorded)),

("is\_read\_from\_cache", \_.getOrElse(IsReadFromCacheFeature, false)),

("get\_initial", \_.getOrElse(GetInitialFeature, false)),

("get\_newer", \_.getOrElse(GetNewerFeature, false)),

("get\_middle", \_.getOrElse(GetMiddleFeature, false)),

("get\_older", \_.getOrElse(GetOlderFeature, false)),

("pull\_to\_refresh", \_.getOrElse(PullToRefreshFeature, false)),

("polling", \_.getOrElse(PollingFeature, false)),

("near\_empty", \_.getOrElse(ServedSizeFeature, None).exists(\_ < 3)),

("is\_request\_context\_launch", \_.getOrElse(IsLaunchRequestFeature, false)),

("mutual\_follow", \_.getOrElse(EarlybirdFeature, None).exists(\_.fromMutualFollow)),

(

"less\_than\_10\_mins\_since\_lnpt",

\_.getOrElse(LastNonPollingTimeFeature, None).exists(\_.untilNow < 10.minutes)),

("served\_in\_conversation\_module", \_.getOrElse(ServedInConversationModuleFeature, false)),

("has\_ticketed\_space", \_.getOrElse(AudioSpaceMetaDataFeature, None).exists(\_.hasTickets)),

("in\_utis\_top5", \_.getOrElse(PositionFeature, None).exists(\_ < 5)),

(

"conversation\_module\_has\_2\_displayed\_tweets",

\_.getOrElse(ConversationModule2DisplayedTweetsFeature, false)),

("empty\_request", \_.getOrElse(ServedSizeFeature, None).exists(\_ == 0)),

("served\_size\_less\_than\_50", \_.getOrElse(ServedSizeFeature, None).exists(\_ < 50)),

(

"served\_size\_between\_50\_and\_100",

\_.getOrElse(ServedSizeFeature, None).exists(size => size >= 50 && size < 100)),

("authored\_by\_contextual\_user", \_.getOrElse(AuthoredByContextualUserFeature, false)),

(

"is\_self\_thread\_tweet",

\_.getOrElse(ConversationFeature, None).exists(\_.isSelfThreadTweet.contains(true))),

("has\_ancestors", \_.getOrElse(AncestorsFeature, Seq.empty).nonEmpty),

("full\_scoring\_succeeded", \_.getOrElse(FullScoringSucceededFeature, false)),

("served\_size\_less\_than\_20", \_.getOrElse(ServedSizeFeature, None).exists(\_ < 20)),

("served\_size\_less\_than\_10", \_.getOrElse(ServedSizeFeature, None).exists(\_ < 10)),

("served\_size\_less\_than\_5", \_.getOrElse(ServedSizeFeature, None).exists(\_ < 5)),

(

"account\_age\_less\_than\_30\_minutes",

\_.getOrElse(AccountAgeFeature, None).exists(\_.untilNow < 30.minutes)),

("conversation\_module\_has\_gap", \_.getOrElse(ConversationModuleHasGapFeature, false)),

(

"directed\_at\_user\_is\_in\_first\_degree",

\_.getOrElse(EarlybirdFeature, None).exists(\_.directedAtUserIdIsInFirstDegree.contains(true))),

(

"has\_semantic\_core\_annotation",

\_.getOrElse(EarlybirdFeature, None).exists(\_.semanticCoreAnnotations.nonEmpty)),

("is\_request\_context\_foreground", \_.getOrElse(IsForegroundRequestFeature, false)),

(

"account\_age\_less\_than\_1\_day",

\_.getOrElse(AccountAgeFeature, None).exists(\_.untilNow < 1.day)),

(

"account\_age\_less\_than\_7\_days",

\_.getOrElse(AccountAgeFeature, None).exists(\_.untilNow < 7.days)),

(

"part\_of\_utt",

\_.getOrElse(EarlybirdFeature, None)

.exists(\_.semanticCoreAnnotations.exists(\_.exists(annotation =>

annotation.domainId == SemanticCoreFeatures.UnifiedTwitterTaxonomy)))),

(

"has\_home\_latest\_request\_past\_week",

\_.getOrElse(FollowingLastNonPollingTimeFeature, None).exists(\_.untilNow < 7.days)),

("is\_utis\_pos0", \_.getOrElse(PositionFeature, None).exists(\_ == 0)),

("is\_utis\_pos1", \_.getOrElse(PositionFeature, None).exists(\_ == 1)),

("is\_utis\_pos2", \_.getOrElse(PositionFeature, None).exists(\_ == 2)),

("is\_utis\_pos3", \_.getOrElse(PositionFeature, None).exists(\_ == 3)),

("is\_utis\_pos4", \_.getOrElse(PositionFeature, None).exists(\_ == 4)),

("is\_random\_tweet", \_.getOrElse(IsRandomTweetFeature, false)),

("has\_random\_tweet\_in\_response", \_.getOrElse(HasRandomTweetFeature, false)),

("is\_random\_tweet\_above\_in\_utis", \_.getOrElse(IsRandomTweetAboveFeature, false)),

(

"has\_ancestor\_authored\_by\_viewer",

candidate =>

candidate

.getOrElse(AncestorsFeature, Seq.empty).exists(ancestor =>

candidate.getOrElse(ViewerIdFeature, 0L) == ancestor.userId)),

("ancestor", \_.getOrElse(IsAncestorCandidateFeature, false)),

(

"deep\_reply",

candidate =>

candidate.getOrElse(InReplyToTweetIdFeature, None).nonEmpty && candidate

.getOrElse(AncestorsFeature, Seq.empty).size > 2),

(

"has\_simcluster\_embeddings",

\_.getOrElse(

SimclustersTweetTopKClustersWithScoresFeature,

Map.empty[String, Double]).nonEmpty),

(

"tweet\_age\_less\_than\_15\_seconds",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow <= 15.seconds)),

(

"less\_than\_1\_hour\_since\_lnpt",

\_.getOrElse(LastNonPollingTimeFeature, None).exists(\_.untilNow < 1.hour)),

("has\_gte\_10\_favs", \_.getOrElse(EarlybirdFeature, None).exists(\_.favCountV2.exists(\_ >= 10))),

(

"device\_language\_matches\_tweet\_language",

candidate =>

candidate.getOrElse(TweetLanguageFeature, None) ==

candidate.getOrElse(DeviceLanguageFeature, None)),

(

"root\_ancestor",

candidate =>

candidate.getOrElse(IsAncestorCandidateFeature, false) && candidate

.getOrElse(InReplyToTweetIdFeature, None).isEmpty),

("question", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasQuestion.contains(true))),

("in\_network", \_.getOrElse(InNetworkFeature, true)),

(

"has\_political\_annotation",

\_.getOrElse(EarlybirdFeature, None).exists(

\_.semanticCoreAnnotations.exists(

\_.exists(annotation =>

SemanticCoreFeatures.PoliticalDomains.contains(annotation.domainId) ||

(annotation.domainId == SemanticCoreFeatures.UnifiedTwitterTaxonomy &&

annotation.entityId == SemanticCoreFeatures.UttPoliticsEntityId))))),

(

"is\_dont\_at\_me\_by\_invitation",

\_.getOrElse(EarlybirdFeature, None).exists(

\_.conversationControl.exists(\_.isInstanceOf[tpt.ConversationControl.ByInvitation]))),

(

"is\_dont\_at\_me\_community",

\_.getOrElse(EarlybirdFeature, None)

.exists(\_.conversationControl.exists(\_.isInstanceOf[tpt.ConversationControl.Community]))),

("has\_zero\_score", \_.getOrElse(ScoreFeature, None).exists(\_ == 0.0)),

(

"is\_followed\_topic\_tweet",

\_.getOrElse(TopicContextFunctionalityTypeFeature, None)

.exists(\_ == BasicTopicContextFunctionalityType)),

(

"is\_recommended\_topic\_tweet",

\_.getOrElse(TopicContextFunctionalityTypeFeature, None)

.exists(\_ == RecommendationTopicContextFunctionalityType)),

("has\_gte\_100\_favs", \_.getOrElse(EarlybirdFeature, None).exists(\_.favCountV2.exists(\_ >= 100))),

("has\_gte\_1k\_favs", \_.getOrElse(EarlybirdFeature, None).exists(\_.favCountV2.exists(\_ >= 1000))),

(

"has\_gte\_10k\_favs",

\_.getOrElse(EarlybirdFeature, None).exists(\_.favCountV2.exists(\_ >= 10000))),

(

"has\_gte\_100k\_favs",

\_.getOrElse(EarlybirdFeature, None).exists(\_.favCountV2.exists(\_ >= 100000))),

("has\_audio\_space", \_.getOrElse(AudioSpaceMetaDataFeature, None).exists(\_.hasSpace)),

("has\_live\_audio\_space", \_.getOrElse(AudioSpaceMetaDataFeature, None).exists(\_.isLive)),

(

"has\_gte\_10\_retweets",

\_.getOrElse(EarlybirdFeature, None).exists(\_.retweetCountV2.exists(\_ >= 10))),

(

"has\_gte\_100\_retweets",

\_.getOrElse(EarlybirdFeature, None).exists(\_.retweetCountV2.exists(\_ >= 100))),

(

"has\_gte\_1k\_retweets",

\_.getOrElse(EarlybirdFeature, None).exists(\_.retweetCountV2.exists(\_ >= 1000))),

(

"has\_us\_political\_annotation",

\_.getOrElse(EarlybirdFeature, None)

.exists(\_.semanticCoreAnnotations.exists(\_.exists(annotation =>

annotation.domainId == SemanticCoreFeatures.UnifiedTwitterTaxonomy &&

annotation.entityId == SemanticCoreFeatures.usPoliticalTweetEntityId &&

annotation.groupId == SemanticCoreFeatures.UsPoliticalTweetAnnotationGroupIds.BalancedV0)))),

(

"has\_toxicity\_score\_above\_threshold",

\_.getOrElse(EarlybirdFeature, None).exists(\_.toxicityScore.exists(\_ > 0.91))),

("is\_topic\_tweet", \_.getOrElse(TopicIdSocialContextFeature, None).isDefined),

(

"text\_only",

candidate =>

candidate.getOrElse(HasDisplayedTextFeature, false) &&

!(candidate.getOrElse(EarlybirdFeature, None).exists(\_.hasImage) ||

candidate.getOrElse(EarlybirdFeature, None).exists(\_.hasVideo) ||

candidate.getOrElse(EarlybirdFeature, None).exists(\_.hasCard))),

(

"image\_only",

candidate =>

candidate.getOrElse(EarlybirdFeature, None).exists(\_.hasImage) &&

!candidate.getOrElse(HasDisplayedTextFeature, false)),

("has\_1\_image", \_.getOrElse(NumImagesFeature, None).exists(\_ == 1)),

("has\_2\_images", \_.getOrElse(NumImagesFeature, None).exists(\_ == 2)),

("has\_3\_images", \_.getOrElse(NumImagesFeature, None).exists(\_ == 3)),

("has\_4\_images", \_.getOrElse(NumImagesFeature, None).exists(\_ == 4)),

("has\_card", \_.getOrElse(EarlybirdFeature, None).exists(\_.hasCard)),

("user\_follow\_count\_gte\_50", \_.getOrElse(UserFollowingCountFeature, None).exists(\_ > 50)),

(

"has\_liked\_by\_social\_context",

candidateFeatures =>

candidateFeatures

.getOrElse(SGSValidLikedByUserIdsFeature, Seq.empty)

.exists(candidateFeatures

.getOrElse(PerspectiveFilteredLikedByUserIdsFeature, Seq.empty).toSet.contains)),

(

"has\_followed\_by\_social\_context",

\_.getOrElse(SGSValidFollowedByUserIdsFeature, Seq.empty).nonEmpty),

(

"has\_topic\_social\_context",

candidateFeatures =>

candidateFeatures

.getOrElse(TopicIdSocialContextFeature, None)

.isDefined &&

candidateFeatures.getOrElse(TopicContextFunctionalityTypeFeature, None).isDefined),

("video\_lte\_10\_sec", \_.getOrElse(VideoDurationMsFeature, None).exists(\_ <= 10000)),

(

"video\_bt\_10\_60\_sec",

\_.getOrElse(VideoDurationMsFeature, None).exists(duration =>

duration > 10000 && duration <= 60000)),

("video\_gt\_60\_sec", \_.getOrElse(VideoDurationMsFeature, None).exists(\_ > 60000)),

(

"tweet\_age\_lte\_30\_minutes",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow <= 30.minutes)),

(

"tweet\_age\_lte\_1\_hour",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow <= 1.hour)),

(

"tweet\_age\_lte\_6\_hours",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow <= 6.hours)),

(

"tweet\_age\_lte\_12\_hours",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow <= 12.hours)),

(

"tweet\_age\_gte\_24\_hours",

\_.getOrElse(OriginalTweetCreationTimeFromSnowflakeFeature, None)

.exists(\_.untilNow >= 24.hours)),

)

val PredicateMap = CandidatePredicates.toMap

}