package com.twitter.cr\_mixer.model

import com.twitter.cr\_mixer.thriftscala.SimilarityEngineType

import com.twitter.cr\_mixer.thriftscala.SourceType

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.util.Time

/\*\*\*

\* Tweet-level attributes. Represents the source used in candidate generation

\* Due to legacy reason, SourceType used to represent both SourceType and SimilarityEngineType

\* Moving forward, SourceType will be used for SourceType ONLY. eg., TweetFavorite, UserFollow, TwiceUserId

\* At the same time, We create a new SimilarityEngineType to separate them. eg., SimClustersANN

\*

\* Currently, one special case is that we have TwiceUserId as a source, which is not necessarily a "signal"

\* @param sourceType, e.g., SourceType.TweetFavorite, SourceType.UserFollow, SourceType.TwiceUserId

\* @param internalId, e.g., UserId(0L), TweetId(0L)

\*/

case class SourceInfo(

sourceType: SourceType,

internalId: InternalId,

sourceEventTime: Option[Time])

/\*\*\*

\* Tweet-level attributes. Represents the source User Graph used in candidate generation

\* It is an intermediate product, and will not be stored, unlike SourceInfo.

\* Essentially, CrMixer queries a graph, and the graph returns a list of users to be used as sources.

\* For instance, RealGraph, EarlyBird, FRS, Stp, etc. The underlying similarity engines such as

\* UTG or UTEG will leverage these sources to build candidates.

\*

\* We extended the definition of SourceType to cover both "Source Signal" and "Source Graph"

\* See [CrMixer] Graph Based Source Fetcher Abstraction Proposal:

\*

\* consider making both SourceInfo and GraphSourceInfo extends the same trait to

\* have a unified interface.

\*/

case class GraphSourceInfo(

sourceType: SourceType,

seedWithScores: Map[UserId, Double])

/\*\*\*

\* Tweet-level attributes. Represents the similarity engine (the algorithm) used for

\* candidate generation along with their metadata.

\* @param similarityEngineType, e.g., SimClustersANN, UserTweetGraph

\* @param modelId. e.g., UserTweetGraphConsumerEmbedding\_ALL\_20210708

\* @param score - a score generated by this sim engine

\*/

case class SimilarityEngineInfo(

similarityEngineType: SimilarityEngineType,

modelId: Option[String], // ModelId can be a None. e.g., UTEG, UnifiedTweetBasedSE. etc

score: Option[Double])

/\*\*\*\*

\* Tweet-level attributes. A combination for both SourceInfo and SimilarityEngineInfo

\* SimilarityEngine is a composition, and it can be composed by many leaf Similarity Engines.

\* For instance, the TweetBasedUnified SE could be a composition of both UserTweetGraph SE, SimClustersANN SE.

\* Note that a SimilarityEngine (Composite) may call other SimilarityEngines (Atomic, Contributing)

\* to contribute to its final candidate list. We track these Contributing SEs in the contributingSimilarityEngines list

\*

\* @param sourceInfoOpt - this is optional as many consumerBased CG does not have a source

\* @param similarityEngineInfo - the similarity engine used in Candidate Generation (eg., TweetBasedUnifiedSE). It can be an atomic SE or an composite SE

\* @param contributingSimilarityEngines - only composite SE will have it (e.g., SANNN, UTG). Otherwise it is an empty Seq. All contributing SEs mst be atomic

\*/

case class CandidateGenerationInfo(

sourceInfoOpt: Option[SourceInfo],

similarityEngineInfo: SimilarityEngineInfo,

contributingSimilarityEngines: Seq[SimilarityEngineInfo])