package com.twitter.timelineranker.common

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.util.Future

/\*\*

\* Transform that explicitly hydrates candidate tweets and fetches source tweets in parallel

\* and then joins the results back into the original Envelope

\* @param candidateTweetHydration Pipeline that hydrates candidate tweets

\* @param sourceTweetHydration Pipeline that fetches and hydrates source tweets

\*/

class HydrateTweetsAndSourceTweetsInParallelTransform(

candidateTweetHydration: FutureArrow[CandidateEnvelope, CandidateEnvelope],

sourceTweetHydration: FutureArrow[CandidateEnvelope, CandidateEnvelope])

extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

override def apply(envelope: CandidateEnvelope): Future[CandidateEnvelope] = {

Future

.join(

candidateTweetHydration(envelope),

sourceTweetHydration(envelope)

).map {

case (candidateTweetEnvelope, sourceTweetEnvelope) =>

envelope.copy(

hydratedTweets = candidateTweetEnvelope.hydratedTweets,

sourceSearchResults = sourceTweetEnvelope.sourceSearchResults,

sourceHydratedTweets = sourceTweetEnvelope.sourceHydratedTweets

)

}

}

}