package com.twitter.graph.batch.job.tweepcred

import com.twitter.pluck.source.combined\_user\_source.MostRecentCombinedUserSnapshotSource

import com.twitter.scalding.\_

/\*\*

\* Calculate tweepcred from the given pagerank file. If post\_adjust is true,

\* reduce pagerank for users with low followers compared to number of

\* followings based on existing reputation code.

\* Options:

\* --input\_pagerank: given pagerank

\* --user\_mass: user mass tsv file, generated by twadoop user\_mass job

\* --output\_pagerank: where to put pagerank file

\* --output\_tweepcred: where to put tweepcred file

\* optional arguments:

\* --post\_adjust: whether to do post adjust, default true

\*

\*/

class ExtractTweepcred(args: Args) extends Job(args) {

val POST\_ADJUST = args.getOrElse("post\_adjust", "true").toBoolean

val inputPagerank = getInputPagerank(args("input\_pagerank"))

.map(() -> ('num\_followers, 'num\_followings)) { (u: Unit) =>

(0, 0)

}

val userInfo = TypedPipe

.from(MostRecentCombinedUserSnapshotSource)

.flatMap { combinedUser =>

val user = Option(combinedUser.user)

val userId = user.map(\_.id).getOrElse(0L)

val userExtended = Option(combinedUser.user\_extended)

val numFollowers = userExtended.flatMap(u => Option(u.followers)).map(\_.toInt).getOrElse(0)

val numFollowings = userExtended.flatMap(u => Option(u.followings)).map(\_.toInt).getOrElse(0)

if (userId == 0L || user.map(\_.safety).exists(\_.deactivated)) {

None

} else {

Some((userId, 0.0, numFollowers, numFollowings))

}

}

.toPipe[(Long, Double, Int, Int)]('src\_id, 'mass\_input, 'num\_followers, 'num\_followings)

val pagerankWithSuspended = (inputPagerank ++ userInfo)

.groupBy('src\_id) {

\_.max('mass\_input)

.max('num\_followers)

.max('num\_followings)

}

pagerankWithSuspended

.discard('num\_followers, 'num\_followings)

.write(Tsv(args("output\_pagerank")))

val adjustedPagerank =

if (POST\_ADJUST) {

pagerankWithSuspended

.map(('mass\_input, 'num\_followers, 'num\_followings) -> 'mass\_input) {

input: (Double, Int, Int) =>

Reputation.adjustReputationsPostCalculation(input.\_1, input.\_2, input.\_3)

}

.normalize('mass\_input)

} else {

pagerankWithSuspended

.discard('num\_followers, 'num\_followings)

}

val tweepcred = adjustedPagerank

.map('mass\_input -> 'mass\_input) { input: Double =>

Reputation.scaledReputation(input)

}

tweepcred.write(Tsv(args("output\_tweepcred")))

tweepcred.write(Tsv(args("current\_tweepcred")))

tweepcred.write(Tsv(args("today\_tweepcred")))

def getInputPagerank(fileName: String) = {

Tsv(fileName).read

.mapTo((0, 1) -> ('src\_id, 'mass\_input)) { input: (Long, Double) =>

input

}

}

}