package com.twitter.search.ingester.pipeline.twitter;

import org.apache.commons.pipeline.StageException;

import org.apache.commons.pipeline.validation.ConsumedTypes;

import org.apache.commons.pipeline.validation.ProducedTypes;

import com.twitter.search.common.decider.DeciderUtil;

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

import com.twitter.search.ingester.model.IngesterTwitterMessage;

import com.twitter.search.ingester.pipeline.util.PipelineStageRuntimeException;

/\*\*

\* Filters out tweets that are not retweets or replies.

\*/

@ConsumedTypes(IngesterTwitterMessage.class)

@ProducedTypes(IngesterTwitterMessage.class)

public class FilterRetweetsAndRepliesStage extends TwitterBaseStage

<IngesterTwitterMessage, IngesterTwitterMessage> {

private static final String EMIT\_RETWEET\_AND\_REPLY\_ENGAGEMENTS\_DECIDER\_KEY =

"ingester\_realtime\_emit\_retweet\_and\_reply\_engagements";

private SearchRateCounter filteredRetweetsCount;

private SearchRateCounter filteredRepliesToTweetsCount;

private SearchRateCounter incomingRetweetsAndRepliesToTweetsCount;

@Override

public void initStats() {

super.initStats();

innerSetupStats();

}

@Override

protected void innerSetupStats() {

filteredRetweetsCount =

SearchRateCounter.export(getStageNamePrefix() + "\_filtered\_retweets\_count");

filteredRepliesToTweetsCount =

SearchRateCounter.export(getStageNamePrefix() + "\_filtered\_replies\_to\_tweets\_count");

incomingRetweetsAndRepliesToTweetsCount =

SearchRateCounter.export(

getStageNamePrefix() + "\_incoming\_retweets\_and\_replies\_to\_tweets\_count");

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterTwitterMessage)) {

throw new StageException(this, "Object is not an IngesterTwitterMessage: " + obj);

}

IngesterTwitterMessage status = (IngesterTwitterMessage) obj;

if (tryToFilter(status)) {

emitAndCount(status);

}

}

@Override

public IngesterTwitterMessage runStageV2(IngesterTwitterMessage message) {

if (!tryToFilter(message)) {

throw new PipelineStageRuntimeException("Does not have to pass to the next stage.");

}

return message;

}

private boolean tryToFilter(IngesterTwitterMessage status) {

boolean shouldEmit = false;

if (status.isRetweet() || status.isReplyToTweet()) {

incomingRetweetsAndRepliesToTweetsCount.increment();

if (DeciderUtil.isAvailableForRandomRecipient(

decider, EMIT\_RETWEET\_AND\_REPLY\_ENGAGEMENTS\_DECIDER\_KEY)) {

if (status.isRetweet()) {

filteredRetweetsCount.increment();

} else {

filteredRepliesToTweetsCount.increment();

}

shouldEmit = true;

}

}

return shouldEmit;

}

}