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.ProducesConsumed;

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

import com.twitter.search.common.relevance.entities.TwitterMessage;

import com.twitter.search.ingester.pipeline.twitter.filters.IngesterValidMessageFilter;

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

/\*\*

\* Filter out Twitter messages meeting some filtering rule.

\*/

@ConsumedTypes(TwitterMessage.class)

@ProducesConsumed

public class FilterTwitterMessageStage extends TwitterBaseStage

<TwitterMessage, TwitterMessage> {

private IngesterValidMessageFilter filter = null;

private SearchRateCounter validMessages;

private SearchRateCounter invalidMessages;

@Override

protected void initStats() {

super.initStats();

innerSetupStats();

}

@Override

protected void innerSetupStats() {

validMessages = SearchRateCounter.export(getStageNamePrefix() + "\_valid\_messages");

invalidMessages = SearchRateCounter.export(getStageNamePrefix() + "\_filtered\_messages");

}

@Override

protected void doInnerPreprocess() {

innerSetup();

}

@Override

protected void innerSetup() {

filter = new IngesterValidMessageFilter(decider);

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof TwitterMessage)) {

throw new StageException(this, "Object is not a IngesterTwitterMessage: "

+ obj);

}

TwitterMessage message = (TwitterMessage) obj;

if (tryToFilter(message)) {

emitAndCount(message);

}

}

@Override

protected TwitterMessage innerRunStageV2(TwitterMessage message) {

if (!tryToFilter(message)) {

throw new PipelineStageRuntimeException("Failed to filter, does not have to "

+ "pass to the next stage");

}

return message;

}

private boolean tryToFilter(TwitterMessage message) {

boolean ableToFilter = false;

if (message != null && filter.accepts(message)) {

validMessages.increment();

ableToFilter = true;

} else {

invalidMessages.increment();

}

return ableToFilter;

}

}