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 org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.indexing.thriftjava.ThriftGeoLocationSource;

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

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

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

import com.twitter.search.common.relevance.text.LocationUtils;

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

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

/\*\*

\* Read-only stage to extract lat/lon pairs from the tweet text and populate

\* the geoLocation field.

\* <p>

\* If the tweet is geotagged by mobile devices, the geo coordinates extracted from the JSON

\* is used.

\*/

@ConsumedTypes(IngesterTwitterMessage.class)

@ProducesConsumed

public class SingleTweetExtractAndGeocodeLatLonStage extends TwitterBaseStage

<TwitterMessage, IngesterTwitterMessage> {

private static final Logger LOG =

LoggerFactory.getLogger(SingleTweetExtractAndGeocodeLatLonStage.class);

private SearchRateCounter extractedLatLons;

private SearchRateCounter badLatLons;

@Override

public void initStats() {

super.initStats();

innerSetupStats();

}

@Override

protected void innerSetupStats() {

extractedLatLons = SearchRateCounter.export(getStageNamePrefix() + "\_extracted\_lat\_lons");

badLatLons = SearchRateCounter.export(getStageNamePrefix() + "\_invalid\_lat\_lons");

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterTwitterMessage)) {

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

}

IngesterTwitterMessage message = IngesterTwitterMessage.class.cast(obj);

tryToSetGeoLocation(message);

emitAndCount(message);

}

@Override

protected IngesterTwitterMessage innerRunStageV2(TwitterMessage message) {

// Previous stage takes in a TwitterMessage and returns a TwitterMessage. I think it was

// done to simplify testing. From this stage onwards, we only count the message that are of type

// IngesterTwitterMessage.

if (!(message instanceof IngesterTwitterMessage)) {

throw new PipelineStageRuntimeException("Message needs to be of type IngesterTwitterMessage");

}

IngesterTwitterMessage ingesterTwitterMessage = IngesterTwitterMessage.class.cast(message);

tryToSetGeoLocation(ingesterTwitterMessage);

return ingesterTwitterMessage;

}

private void tryToSetGeoLocation(IngesterTwitterMessage message) {

if (message.getGeoTaggedLocation() != null) {

message.setGeoLocation(message.getGeoTaggedLocation());

} else if (message.hasGeoLocation()) {

LOG.warn("Message {} already contains geoLocation", message.getId());

} else {

try {

GeoObject extracted = extractLatLon(message);

if (extracted != null) {

message.setGeoLocation(extracted);

extractedLatLons.increment();

}

} catch (NumberFormatException e) {

LOG.debug("Message contains bad latitude and longitude: " + message.getOrigLocation(), e);

badLatLons.increment();

} catch (Exception e) {

LOG.error("Failed to extract geo location from " + message.getOrigLocation() + " for tweet "

+ message.getId(), e);

}

}

}

private GeoObject extractLatLon(IngesterTwitterMessage message) throws NumberFormatException {

double[] latlon = LocationUtils.extractLatLon(message);

return latlon == null

? null

: new GeoObject(latlon[0], latlon[1], ThriftGeoLocationSource.TWEET\_TEXT);

}

}