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

import java.util.Set;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.collect.Sets;

import org.apache.commons.lang.StringUtils;

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.decider.DeciderUtil;

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

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

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

@ConsumedTypes(TwitterMessage.class)

@ProducesConsumed

public class RetrieveSpaceIdsStage extends TwitterBaseStage

<TwitterMessage, TwitterMessage> {

@VisibleForTesting

protected static final Pattern SPACES\_URL\_REGEX =

Pattern.compile("^https://twitter\\.com/i/spaces/([a-zA-Z0-9]+)\\S\*$");

@VisibleForTesting

protected static final String PARSE\_SPACE\_ID\_DECIDER\_KEY = "ingester\_all\_parse\_space\_id\_from\_url";

private static SearchRateCounter numTweetsWithSpaceIds;

private static SearchRateCounter numTweetsWithMultipleSpaceIds;

@Override

protected void initStats() {

super.initStats();

innerSetupStats();

}

@Override

protected void innerSetupStats() {

numTweetsWithSpaceIds = SearchRateCounter.export(

getStageNamePrefix() + "\_tweets\_with\_space\_ids");

numTweetsWithMultipleSpaceIds = SearchRateCounter.export(

getStageNamePrefix() + "\_tweets\_with\_multiple\_space\_ids");

}

@Override

public void innerProcess(Object obj) throws StageException {

TwitterMessage message = (TwitterMessage) obj;

tryToRetrieveSpaceId(message);

emitAndCount(message);

}

private void tryToRetrieveSpaceId(TwitterMessage message) {

if (DeciderUtil.isAvailableForRandomRecipient(decider, PARSE\_SPACE\_ID\_DECIDER\_KEY)) {

Set<String> spaceIds = parseSpaceIdsFromMessage(message);

int spaceIdCount = spaceIds.size();

if (spaceIdCount > 0) {

numTweetsWithSpaceIds.increment();

if (spaceIdCount > 1) {

numTweetsWithMultipleSpaceIds.increment();

}

message.setSpaceIds(spaceIds);

}

}

}

@Override

protected TwitterMessage innerRunStageV2(TwitterMessage message) {

tryToRetrieveSpaceId(message);

return message;

}

private String parseSpaceIdsFromUrl(String url) {

String spaceId = null;

if (StringUtils.isNotEmpty(url)) {

Matcher matcher = SPACES\_URL\_REGEX.matcher(url);

if (matcher.matches()) {

spaceId = matcher.group(1);

}

}

return spaceId;

}

private Set<String> parseSpaceIdsFromMessage(TwitterMessage message) {

Set<String> spaceIds = Sets.newHashSet();

for (ThriftExpandedUrl expandedUrl : message.getExpandedUrls()) {

String spaceId = parseSpaceIdsFromUrl(expandedUrl.getExpandedUrl());

if (StringUtils.isNotEmpty(spaceId)) {

spaceIds.add(spaceId);

}

}

return spaceIds;

}

}