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

import java.util.Set;

import scala.Option;

import com.google.common.collect.ImmutableSet;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.cuad.ner.plain.thriftjava.NamedEntities;

import com.twitter.cuad.ner.plain.thriftjava.NamedEntity;

import com.twitter.decider.Decider;

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.strato\_fetchers.NamedEntityFetcher;

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

import com.twitter.strato.catalog.Fetch;

import com.twitter.util.Future;

/\*\*

\* Handles the retrieval and population of named entities in TwitterMessages performed

\* by ingesters.

\*/

class NamedEntityHandler {

private static final Logger LOG = LoggerFactory.getLogger(NamedEntityHandler.class);

private static final String RETRIEVE\_NAMED\_ENTITIES\_DECIDER\_KEY =

"ingester\_all\_retrieve\_named\_entities\_%s";

// Named entities are only extracted in English, Spanish, and Japanese

private static final Set<String> NAMED\_ENTITY\_LANGUAGES = ImmutableSet.of("en", "es", "ja");

private final NamedEntityFetcher namedEntityFetcher;

private final Decider decider;

private final String deciderKey;

private SearchRateCounter lookupStat;

private SearchRateCounter successStat;

private SearchRateCounter namedEntityCountStat;

private SearchRateCounter errorStat;

private SearchRateCounter emptyResponseStat;

private SearchRateCounter deciderSkippedStat;

private IngesterStageTimer retrieveNamedEntitiesTimer;

NamedEntityHandler(

NamedEntityFetcher namedEntityFetcher, Decider decider, String statsPrefix,

String deciderSuffix) {

this.namedEntityFetcher = namedEntityFetcher;

this.decider = decider;

this.deciderKey = String.format(RETRIEVE\_NAMED\_ENTITIES\_DECIDER\_KEY, deciderSuffix);

lookupStat = SearchRateCounter.export(statsPrefix + "\_lookups");

successStat = SearchRateCounter.export(statsPrefix + "\_success");

namedEntityCountStat = SearchRateCounter.export(statsPrefix + "\_named\_entity\_count");

errorStat = SearchRateCounter.export(statsPrefix + "\_error");

emptyResponseStat = SearchRateCounter.export(statsPrefix + "\_empty\_response");

deciderSkippedStat = SearchRateCounter.export(statsPrefix + "\_decider\_skipped");

retrieveNamedEntitiesTimer = new IngesterStageTimer(statsPrefix + "\_request\_timer");

}

Future<Fetch.Result<NamedEntities>> retrieve(IngesterTwitterMessage message) {

lookupStat.increment();

return namedEntityFetcher.fetch(message.getTweetId());

}

void addEntitiesToMessage(IngesterTwitterMessage message, Fetch.Result<NamedEntities> result) {

retrieveNamedEntitiesTimer.start();

Option<NamedEntities> response = result.v();

if (response.isDefined()) {

successStat.increment();

for (NamedEntity namedEntity : response.get().getEntities()) {

namedEntityCountStat.increment();

message.addNamedEntity(namedEntity);

}

} else {

emptyResponseStat.increment();

LOG.debug("Empty NERResponse for named entity query on tweet {}", message.getId());

}

retrieveNamedEntitiesTimer.stop();

}

void incrementErrorCount() {

errorStat.increment();

}

boolean shouldRetrieve(IngesterTwitterMessage message) {

// Use decider to control retrieval of named entities. This allows us to shut off retrieval

// if it causes problems.

if (!DeciderUtil.isAvailableForRandomRecipient(decider, deciderKey)) {

deciderSkippedStat.increment();

return false;

}

// Named entities are only extracted in certain languages, so we can skip tweets

// in other languages

return NAMED\_ENTITY\_LANGUAGES.contains(message.getLanguage());

}

}