package com.twitter.search.earlybird.document;

import com.twitter.common.text.token.TokenProcessor;

import com.twitter.common.text.token.TwitterTokenStream;

import com.twitter.decider.Decider;

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

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

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

import com.twitter.search.common.schema.SchemaDocumentFactory;

import com.twitter.search.common.schema.base.Schema;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

public class TruncationTokenStreamWriter implements SchemaDocumentFactory.TokenStreamRewriter {

private static final int NEVER\_TRUNCATE\_CHARS\_BELOW\_POSITION = 140;

private static final String TRUNCATE\_LONG\_TWEETS\_DECIDER\_KEY\_PREFIX =

"truncate\_long\_tweets\_in\_";

private static final String NUM\_TWEET\_CHARACTERS\_SUPPORTED\_DECIDER\_KEY\_PREFIX =

"num\_tweet\_characters\_supported\_in\_";

private static final SearchCounter NUM\_TWEETS\_TRUNCATED =

SearchCounter.export("num\_tweets\_truncated");

private static final SearchLongGauge NUM\_TWEET\_CHARACTERS\_SUPPORTED =

SearchLongGauge.export("num\_tweet\_characters\_supported");

private final Decider decider;

private final String truncateLongTweetsDeciderKey;

private final String numCharsSupportedDeciderKey;

/\*\*

\* Creates a TruncationTokenStreamWriter

\*/

public TruncationTokenStreamWriter(EarlybirdCluster cluster, Decider decider) {

this.decider = decider;

this.truncateLongTweetsDeciderKey =

TRUNCATE\_LONG\_TWEETS\_DECIDER\_KEY\_PREFIX + cluster.name().toLowerCase();

this.numCharsSupportedDeciderKey =

NUM\_TWEET\_CHARACTERS\_SUPPORTED\_DECIDER\_KEY\_PREFIX + cluster.name().toLowerCase();

}

@Override

public TwitterTokenStream rewrite(Schema.FieldInfo fieldInfo, TwitterTokenStream stream) {

if (EarlybirdFieldConstant.TEXT\_FIELD.getFieldName().equals(fieldInfo.getName())) {

final int maxPosition = getTruncatePosition();

NUM\_TWEET\_CHARACTERS\_SUPPORTED.set(maxPosition);

if (maxPosition >= NEVER\_TRUNCATE\_CHARS\_BELOW\_POSITION) {

return new TokenProcessor(stream) {

@Override

public final boolean incrementToken() {

if (incrementInputStream()) {

if (offset() < maxPosition) {

return true;

}

NUM\_TWEETS\_TRUNCATED.increment();

}

return false;

}

};

}

}

return stream;

}

/\*\*

\* Get the truncation position.

\*

\* @return the truncation position or -1 if truncation is disabled.

\*/

private int getTruncatePosition() {

int maxPosition;

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

return -1;

}

maxPosition = DeciderUtil.getAvailability(decider, numCharsSupportedDeciderKey);

if (maxPosition < NEVER\_TRUNCATE\_CHARS\_BELOW\_POSITION) {

// Never truncate below NEVER\_TRUNCATE\_CHARS\_BELOW\_POSITION chars

maxPosition = NEVER\_TRUNCATE\_CHARS\_BELOW\_POSITION;

}

return maxPosition;

}

}