package com.twitter.search.earlybird.document;

import java.io.IOException;

import java.util.List;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Preconditions;

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

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

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

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

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

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

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

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

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

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

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

import com.twitter.search.common.schema.thriftjava.ThriftDocument;

import com.twitter.search.common.schema.thriftjava.ThriftField;

import geo.google.datamodel.GeoAddressAccuracy;

/\*\*

\* Used to preprocess a ThriftDocument before indexing.

\*/

public final class ThriftDocumentPreprocessor {

private static final FieldNameToIdMapping ID\_MAP = new EarlybirdFieldConstants();

private static final String FILTER\_LINK\_VALUE = EarlybirdThriftDocumentUtil.formatFilter(

EarlybirdFieldConstant.LINKS\_FIELD.getFieldName());

private static final String HAS\_LINK\_VALUE = EarlybirdFieldConstant.getFacetSkipFieldName(

EarlybirdFieldConstant.LINKS\_FIELD.getFieldName());

private ThriftDocumentPreprocessor() {

}

/\*\*

\* Processes the given document.

\*/

public static ThriftDocument preprocess(

ThriftDocument doc, EarlybirdCluster cluster, ImmutableSchemaInterface schema)

throws IOException {

patchArchiveThriftDocumentAccuracy(doc, cluster);

patchArchiveHasLinks(doc, cluster);

addAllMissingMinEngagementFields(doc, cluster, schema);

return doc;

}

private static final SearchCounter GEO\_SCRUBBED\_COUNT =

SearchCounter.export("geo\_scrubbed\_count");

private static final SearchCounter GEO\_ARCHIVE\_PATCHED\_ACCURACY\_COUNT =

SearchCounter.export("geo\_archive\_patched\_accuracy\_count");

private static final SearchCounter GEO\_MISSING\_COORDINATE\_COUNT =

SearchCounter.export("geo\_missing\_coordinate\_count");

private static final SearchCounter ARCHIVED\_LINKS\_FIELD\_PATCHED\_COUNT =

SearchCounter.export("links\_field\_patched\_count");

/\*\*

\* Counter for all the combinations of nullcast bit set and nullcast filter set.

\*

\* Sum over `ThriftDocumentPreprocessor\_nullcast\_doc\_stats\_\_nullcastBitSet\_true\_\*` to get all docs

\* with nullcast bit set to true.

\*/

private static final SearchTruthTableCounter NULLCAST\_DOC\_STATS =

SearchTruthTableCounter.export(

"ThriftDocumentPreprocessor\_nullcast\_doc\_stats",

"nullcastBitSet",

"nullcastFilterSet");

/\*\*\*

\* See JIRA SEARCH-7329

\*/

private static void patchArchiveThriftDocumentAccuracy(ThriftDocument doc,

EarlybirdCluster cluster) {

ThriftField geoField = ThriftDocumentUtil.getField(

doc,

EarlybirdFieldConstant.GEO\_HASH\_FIELD.getFieldName(),

ID\_MAP);

if (geoField != null) {

if (!geoField.getFieldData().isSetGeoCoordinate()) {

GEO\_MISSING\_COORDINATE\_COUNT.increment();

return;

}

// -1 means that the data is geo scrubbed.

if (geoField.getFieldData().getGeoCoordinate().getAccuracy() == -1) {

doc.getFields().remove(geoField);

GEO\_SCRUBBED\_COUNT.increment();

} else if (EarlybirdCluster.isArchive(cluster)) {

// In archive indexing, we base precision on SearchArchiveStatus.getPrecision, which is not

// in the scale we want. We always use POINT\_LEVEL scale for now.

geoField.getFieldData().getGeoCoordinate().setAccuracy(

GeoAddressAccuracy.POINT\_LEVEL.getCode());

GEO\_ARCHIVE\_PATCHED\_ACCURACY\_COUNT.increment();

}

}

}

/\*\*

\* See SEARCH-9635

\* This patch is used to replace

\* ("field":"internal","term":"\_\_filter\_links") with

\* ("field":"internal","term":"\_\_has\_links").

\*/

private static void patchArchiveHasLinks(ThriftDocument doc, EarlybirdCluster cluster) {

if (!EarlybirdCluster.isArchive(cluster)) {

return;

}

List<ThriftField> fieldList = ThriftDocumentUtil.getFields(doc,

EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName(),

ID\_MAP);

for (ThriftField field : fieldList) {

if (field.getFieldData().getStringValue().equals(FILTER\_LINK\_VALUE)) {

field.getFieldData().setStringValue(HAS\_LINK\_VALUE);

ARCHIVED\_LINKS\_FIELD\_PATCHED\_COUNT.increment();

break;

}

}

}

/\*\*

\* Check whether the nullcast bit and nullcast filter are consistent in the given doc.

\*/

public static boolean isNullcastBitAndFilterConsistent(ThriftDocument doc,

ImmutableSchemaInterface schema) {

return isNullcastBitAndFilterConsistent(doc, schema, NULLCAST\_DOC\_STATS);

}

@VisibleForTesting

static boolean isNullcastBitAndFilterConsistent(

ThriftDocument doc, ImmutableSchemaInterface schema, SearchTruthTableCounter nullCastStats) {

final boolean isNullcastBitSet = EarlybirdThriftDocumentUtil.isNullcastBitSet(schema, doc);

final boolean isNullcastFilterSet = EarlybirdThriftDocumentUtil.isNullcastFilterSet(doc);

// Track stats.

nullCastStats.record(isNullcastBitSet, isNullcastFilterSet);

return isNullcastBitSet == isNullcastFilterSet;

}

@VisibleForTesting

static void addAllMissingMinEngagementFields(

ThriftDocument doc, EarlybirdCluster cluster, ImmutableSchemaInterface schema

) throws IOException {

if (!EarlybirdCluster.isArchive(cluster)) {

return;

}

EarlybirdFieldConstants.EarlybirdFieldConstant encodedFeatureFieldConstant =

EarlybirdFieldConstant.ENCODED\_TWEET\_FEATURES\_FIELD;

byte[] encodedFeaturesBytes = ThriftDocumentUtil.getBytesValue(doc,

encodedFeatureFieldConstant.getFieldName(), ID\_MAP);

if (encodedFeaturesBytes == null) {

return;

}

EarlybirdEncodedFeatures encodedFeatures = EarlybirdEncodedFeaturesUtil.fromBytes(

schema,

EarlybirdFieldConstant.ENCODED\_TWEET\_FEATURES\_FIELD,

encodedFeaturesBytes,

0);

for (String field: EarlybirdFieldConstants.MIN\_ENGAGEMENT\_FIELD\_TO\_CSF\_NAME\_MAP.keySet()) {

EarlybirdFieldConstant csfEngagementField = EarlybirdFieldConstants

.MIN\_ENGAGEMENT\_FIELD\_TO\_CSF\_NAME\_MAP.get(field);

Preconditions.checkState(csfEngagementField != null);

int engagementCounter = encodedFeatures.getFeatureValue(csfEngagementField);

EarlybirdThriftDocumentUtil.addNormalizedMinEngagementField(doc, field, engagementCounter);

}

}

}