package com.twitter.search.common.converter.earlybird;

import java.io.IOException;

import java.util.List;

import javax.annotation.concurrent.NotThreadSafe;

import com.google.common.base.Preconditions;

import com.twitter.common\_internal.text.version.PenguinVersion;

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

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

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

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

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

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

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

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

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

/\*\*

\* CombinedIndexingConverter builds objects from TwitterMessage to ThriftVersionedEvent.

\*

\* It is used in tests and in offline jobs, so all data is available on the TwitterMessage. This

\* means that we don't need to split up the ThriftVersionedEvents into basic events and update

\* events, like we do in the realtime pipeline using the BasicIndexingConverter and the

\* DelayedIndexingConverter.

\*/

@NotThreadSafe

public class CombinedIndexingConverter {

private final EncodedFeatureBuilder featureBuilder;

private final Schema schema;

private final EarlybirdCluster cluster;

public CombinedIndexingConverter(Schema schema, EarlybirdCluster cluster) {

this.featureBuilder = new EncodedFeatureBuilder();

this.schema = schema;

this.cluster = cluster;

}

/\*\*

\* Converts a TwitterMessage to a Thrift representation.

\*/

public ThriftVersionedEvents convertMessageToThrift(

TwitterMessage message,

boolean strict,

List<PenguinVersion> penguinVersions) throws IOException {

Preconditions.checkNotNull(message);

Preconditions.checkNotNull(penguinVersions);

ThriftVersionedEvents versionedEvents = new ThriftVersionedEvents()

.setId(message.getId());

ImmutableSchemaInterface schemaSnapshot = schema.getSchemaSnapshot();

for (PenguinVersion penguinVersion : penguinVersions) {

ThriftDocument document =

buildDocumentForPenguinVersion(schemaSnapshot, message, strict, penguinVersion);

ThriftIndexingEvent thriftIndexingEvent = new ThriftIndexingEvent()

.setDocument(document)

.setEventType(ThriftIndexingEventType.INSERT)

.setSortId(message.getId());

message.getFromUserTwitterId().map(thriftIndexingEvent::setUid);

versionedEvents.putToVersionedEvents(penguinVersion.getByteValue(), thriftIndexingEvent);

}

return versionedEvents;

}

private ThriftDocument buildDocumentForPenguinVersion(

ImmutableSchemaInterface schemaSnapshot,

TwitterMessage message,

boolean strict,

PenguinVersion penguinVersion) throws IOException {

EncodedFeatureBuilder.TweetFeatureWithEncodeFeatures tweetFeature =

featureBuilder.createTweetFeaturesFromTwitterMessage(

message, penguinVersion, schemaSnapshot);

EarlybirdThriftDocumentBuilder builder =

BasicIndexingConverter.buildBasicFields(message, schemaSnapshot, cluster, tweetFeature);

BasicIndexingConverter

.buildUserFields(builder, message, tweetFeature.versionedFeatures, penguinVersion);

BasicIndexingConverter.buildGeoFields(builder, message, tweetFeature.versionedFeatures);

DelayedIndexingConverter.buildURLFields(builder, message, tweetFeature.encodedFeatures);

BasicIndexingConverter.buildRetweetAndReplyFields(builder, message, strict);

BasicIndexingConverter.buildQuotesFields(builder, message);

BasicIndexingConverter.buildVersionedFeatureFields(builder, tweetFeature.versionedFeatures);

DelayedIndexingConverter.buildCardFields(builder, message, penguinVersion);

BasicIndexingConverter.buildAnnotationFields(builder, message);

BasicIndexingConverter.buildNormalizedMinEngagementFields(

builder, tweetFeature.encodedFeatures, cluster);

DelayedIndexingConverter.buildNamedEntityFields(builder, message);

BasicIndexingConverter.buildDirectedAtFields(builder, message);

return builder.build();

}

}