package com.twitter.search.core.earlybird.facets;

import org.apache.lucene.util.BytesRef;

import com.twitter.search.common.hashtable.HashTable;

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

import com.twitter.search.common.util.analysis.IntTermAttributeImpl;

import com.twitter.search.common.util.analysis.LongTermAttributeImpl;

import com.twitter.search.common.util.analysis.SortableLongTermAttributeImpl;

import com.twitter.search.core.earlybird.index.inverted.InvertedIndex;

/\*\*

\* Given a termID this accessor can be used to retrieve the term bytesref and text

\* that corresponds to the termID.

\*/

public interface FacetLabelProvider {

/\*\*

\* Returns a {@link FacetLabelAccessor} for this provider.

\*/

FacetLabelAccessor getLabelAccessor();

abstract class FacetLabelAccessor {

private int currentTermID = -1;

protected final BytesRef termRef = new BytesRef();

protected boolean hasTermPayload = false;

protected final BytesRef termPayload = new BytesRef();

protected int offensiveCount = 0;

protected final boolean maybeSeek(long termID) {

if (termID == currentTermID) {

return true;

}

if (seek(termID)) {

currentTermID = (int) termID;

return true;

} else {

currentTermID = -1;

return false;

}

}

// Seek to term id provided. Returns true if term found. Should update termRef,

// hasTermPayload, and termPayload as appropriate.

protected abstract boolean seek(long termID);

public final BytesRef getTermRef(long termID) {

return maybeSeek(termID) ? termRef : null;

}

public String getTermText(long termID) {

return maybeSeek(termID) ? termRef.utf8ToString() : null;

}

public final BytesRef getTermPayload(long termID) {

return maybeSeek(termID) && hasTermPayload ? termPayload : null;

}

public final int getOffensiveCount(long termID) {

return maybeSeek(termID) ? offensiveCount : 0;

}

}

/\*\*

\* Assumes the term is stored as an IntTermAttribute, and uses this to convert

\* the term bytesref to an integer string facet label.

\*/

class IntTermFacetLabelProvider implements FacetLabelProvider {

private final InvertedIndex invertedIndex;

public IntTermFacetLabelProvider(InvertedIndex invertedIndex) {

this.invertedIndex = invertedIndex;

}

@Override

public FacetLabelAccessor getLabelAccessor() {

return new FacetLabelAccessor() {

@Override

protected boolean seek(long termID) {

if (termID != HashTable.EMPTY\_SLOT) {

invertedIndex.getTerm((int) termID, termRef);

return true;

}

return false;

}

@Override

public String getTermText(long termID) {

return maybeSeek(termID)

? Integer.toString(IntTermAttributeImpl.copyBytesRefToInt(termRef))

: null;

}

};

}

}

/\*\*

\* Assumes the term is stored as an LongTermAttribute, and uses this to convert

\* the term bytesref to an long string facet label.

\*/

class LongTermFacetLabelProvider implements FacetLabelProvider {

private final InvertedIndex invertedIndex;

public LongTermFacetLabelProvider(InvertedIndex invertedIndex) {

this.invertedIndex = invertedIndex;

}

@Override

public FacetLabelAccessor getLabelAccessor() {

return new FacetLabelAccessor() {

@Override

protected boolean seek(long termID) {

if (termID != HashTable.EMPTY\_SLOT) {

invertedIndex.getTerm((int) termID, termRef);

return true;

}

return false;

}

@Override

public String getTermText(long termID) {

return maybeSeek(termID)

? Long.toString(LongTermAttributeImpl.copyBytesRefToLong(termRef))

: null;

}

};

}

}

class SortedLongTermFacetLabelProvider implements FacetLabelProvider {

private final InvertedIndex invertedIndex;

public SortedLongTermFacetLabelProvider(InvertedIndex invertedIndex) {

this.invertedIndex = invertedIndex;

}

@Override

public FacetLabelAccessor getLabelAccessor() {

return new FacetLabelAccessor() {

@Override

protected boolean seek(long termID) {

if (termID != HashTable.EMPTY\_SLOT) {

invertedIndex.getTerm((int) termID, termRef);

return true;

}

return false;

}

@Override

public String getTermText(long termID) {

return maybeSeek(termID)

? Long.toString(SortableLongTermAttributeImpl.copyBytesRefToLong(termRef))

: null;

}

};

}

}

class IdentityFacetLabelProvider implements FacetLabelProvider {

@Override

public FacetLabelAccessor getLabelAccessor() {

return new FacetLabelAccessor() {

@Override

protected boolean seek(long termID) {

return true;

}

@Override

public String getTermText(long termID) {

return Long.toString(termID);

}

};

}

}

/\*\*

\* The methods on this provider should NOT be called under normal circumstances!

\*

\* When a facet misses inverted index and does not use CSF, this InaccessibleFacetLabelProvider

\* will be used as a dummy provider. Then, unexptectedFacetLabelAccess counter will be

\* incremented when this provider is used later.

\*

\* Also see:

\* {@link FacetUtil}

\*/

class InaccessibleFacetLabelProvider implements FacetLabelProvider {

private final SearchCounter unexptectedFacetLabelAccess;

public InaccessibleFacetLabelProvider(String fieldName) {

this.unexptectedFacetLabelAccess =

SearchCounter.export("unexpected\_facet\_label\_access\_for\_field\_" + fieldName);

}

@Override

public FacetLabelAccessor getLabelAccessor() {

return new FacetLabelAccessor() {

@Override

protected boolean seek(long termID) {

unexptectedFacetLabelAccess.increment();

return false;

}

};

}

}

}