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

import javax.annotation.Nullable;

import com.google.common.base.Preconditions;

import com.google.common.collect.Iterables;

import org.apache.commons.lang.StringUtils;

import com.twitter.pink\_floyd.thrift.FetchStatusCode;

import com.twitter.pink\_floyd.thrift.HtmlBasics;

import com.twitter.pink\_floyd.thrift.Resolution;

import com.twitter.pink\_floyd.thrift.UrlData;

import com.twitter.service.spiderduck.gen.LinkCategory;

import com.twitter.service.spiderduck.gen.MediaTypes;

import com.twitter.spiderduck.common.URLUtils;

// Helper class with UrlInfo helper functions

public final class ResolveCompressedUrlsUtils {

private ResolveCompressedUrlsUtils() { }

static class UrlInfo {

public String originalUrl;

@Nullable public String resolvedUrl;

@Nullable public String language;

@Nullable public MediaTypes mediaType;

@Nullable public LinkCategory linkCategory;

@Nullable public String description;

@Nullable public String title;

}

/\*\*

\* Determines if the given UrlData instance is fully resolved.

\*

\* Based on discussions with the URL services team, we decided that the most correct way to

\* determine that a URL was fully resolved is to look at a few response fields:

\* - urlDirectInfo: both the media type and link category must be set.

\* - htmlBasics: Pink has successfully parsed the resolved link's metadata.

\* - resolution: Pink was able to successfully get to the last hop in the redirect chain.

\* This is especially important, because some sites have a robots.txt file, which

\* prevents Pink from following the redirect chain once it gets to that site.

\* In that case, we end up with a "last hop" URL, but the FetchStatusCode is not

\* set to OK. We need to ignore these URLs because we don't know if they're really

\* the last hop URLs.

\* Also, Pink has some restrictions on the page size. For example, it does not

\* parse text pages that are larger than 2MB. So if the redirect chain leads Pink

\* to one of these pages, it will stop there. And again, we don't know if this is

\* the last hop URL or not, so we have to ignore that URL.

\*

\* @param urlData The UrlData instance.

\* @return true if the URL data is fully resolved; false otherwise.

\*/

public static boolean isResolved(UrlData urlData) {

// Make sure the mediaType and linkCategory fields are set.

boolean isInfoReady = urlData.isSetUrlDirectInfo()

&& urlData.getUrlDirectInfo().isSetMediaType()

&& urlData.getUrlDirectInfo().isSetLinkCategory();

// The individual HtmlBasics fields might or might not be set, depending on each website.

// However, all fields should be set at the same time, if they are present. Consider the

// resolution complete if at least one of the title, description or language fields is set.

boolean isHtmlReady = urlData.isSetHtmlBasics()

&& (StringUtils.isNotEmpty(urlData.getHtmlBasics().getTitle())

|| StringUtils.isNotEmpty(urlData.getHtmlBasics().getDescription())

|| StringUtils.isNotEmpty(urlData.getHtmlBasics().getLang()));

Resolution resolution = urlData.getResolution();

boolean isResolutionReady = urlData.isSetResolution()

&& StringUtils.isNotEmpty(resolution.getLastHopCanonicalUrl())

&& resolution.getStatus() == FetchStatusCode.OK

&& resolution.getLastHopHttpResponseStatusCode() == 200;

return isHtmlReady && isInfoReady && isResolutionReady;

}

/\*\*

\* Creates a UrlInfo instance from the given URL data.

\*

\* @param urlData urlData from a resolver response.

\* @return the UrlInfo instance.

\*/

public static UrlInfo getUrlInfo(UrlData urlData) {

Preconditions.checkArgument(urlData.isSetResolution());

UrlInfo urlInfo = new UrlInfo();

urlInfo.originalUrl = urlData.url;

Resolution resolution = urlData.getResolution();

if (resolution.isSetLastHopCanonicalUrl()) {

urlInfo.resolvedUrl = resolution.lastHopCanonicalUrl;

} else {

// Just in case lastHopCanonicalUrl is not available (which shouldn't happen)

if (resolution.isSetRedirectionChain()) {

urlInfo.resolvedUrl = Iterables.getLast(resolution.redirectionChain);

} else {

urlInfo.resolvedUrl = urlData.url;

}

urlInfo.resolvedUrl = URLUtils.canonicalizeUrl(urlInfo.resolvedUrl);

}

if (urlData.isSetUrlDirectInfo()) {

urlInfo.mediaType = urlData.urlDirectInfo.mediaType;

urlInfo.linkCategory = urlData.urlDirectInfo.linkCategory;

}

if (urlData.isSetHtmlBasics()) {

HtmlBasics htmlBasics = urlData.getHtmlBasics();

urlInfo.language = htmlBasics.getLang();

if (htmlBasics.isSetDescription()) {

urlInfo.description = htmlBasics.getDescription();

}

if (htmlBasics.isSetTitle()) {

urlInfo.title = htmlBasics.getTitle();

}

}

return urlInfo;

}

}