ChatGPT

Debugging the Google News Scraper Issue

Issue Overview

The logs indicate that **content scraping is still failing for all articles**, even after implementing earlier suggestions. Specifically, every attempt to scrape a Google News RSS article resulted in errors and no content extracted. The pipeline processed 50 articles but extracted **0 successful articles**, as shown by the `Success rate: 0.0%` in the log. This means the fixes have not resolved the core issue yet.

Analysis of the Logs

From the provided output snippet, we see a repeating pattern for each article (articles 40 through 50 in the log):

- The URL being scraped is a Google News redirect (e.g. `https://news.google.com/rss/articles/CBMi...`).
- The system attempts to **decode the Google News redirect**. The log shows messages like:

```
resolve_google_news_url:204 - Attempting to decode Google News redirect:
https://news.google.com/rss/articles/CBMi_gFBVV95cUxNOXByQm5mUzg3...
news_pipeline.google_news_decoder - INFO - decode_url:341 - Successfully
decoded using HTML/API method
resolve_google_news_url:210 - Successfully decoded Google News URL:
https://www.gstatic.com/_/boq-identity/_/r/
```

  It claims a "successful decode," but the **decoded URL is** `https://www.gstatic.com/_/boq-identity/_/r/`, which is clearly not the actual news article domain. In fact, `boq-identity` is related to Google's identity/login service, not the news site. This suggests the decoder **did not retrieve the real article URL** – it likely got stuck at a Google login or consent page.

- After getting that incorrect URL, the scraper tries to fetch content:
- **Trafilatura** (the HTML extraction library) reports a 404 error for that `gstatic` URL (since there's no real content there).
- Then the **MCP Playwright agent** is invoked. Each time, it logs an error: `Error running query: Connection closed` followed by `MCP extraction failed for https://www.gstatic.com/_/boq-identity/_/r/: Connection closed`. In other words, the headless browser/LLM agent could not retrieve anything from that URL either – likely because it's not a valid page or required a login. The "Connection closed" errors suggest the browser session might have been cut off or the OpenAI agent call failed due to the bad URL.

- This repeats for every Google News article, and none succeed. Finally, the scraper logs:

```
Scraping complete: {'processed': 50, 'extracted': 0, 'trafilatura': 0,
'playwright': 0, 'failed': 50, ...}
```

So all 50 articles failed to extract content, resulting in no summaries in Step 4.

Key takeaway: The **Google News redirect decoding is still not working** – it's yielding a gstatic URL (likely a Google sign-in page) instead of the original news article URL. This is why all extraction attempts failed.

Why the Google News Decoding Fails

Google has recently changed how their News RSS redirect URLs work. In the past, one could base64-decode the `.../rss/articles/` link to get the original URL, but **as of mid-2024 this no longer works**. The encoded string is now a **server-side identifier** rather than a direct URL encoding [1]. In practical terms, that means **you cannot obtain the real article link just by decoding the string** on your end.

In the logs, the decoder's "HTML/API method" likely attempted to fetch or resolve the Google link and got redirected to a Google account page (`boq-identity`). This implies Google might require an authorized session or specific approach to get the actual link. Without proper handling, any direct request ends up at a dead-end (hence the `gstatic.com` URL and 404s).

In summary, **the Google News feed URLs inherently cause redirect loops or identity checks**. Your log confirms what our research suggested: trying to decode or fetch these links without Google's own mechanism leads to failures.

Verification of Implemented Fixes

You mentioned implementing suggestions (possibly from the earlier plan). One critical suggestion was to **skip Google News redirect URLs entirely** to avoid this problem [2]. However, the log shows the code still *attempted to decode* those URLs instead of skipping. We do **not** see the warning `Skipping Google News redirect URL...` that the fix was supposed to log. Instead, we see "Attempting to decode..." for each article. This means one of two things:

- The skip logic was not actually applied (perhaps the code wasn't updated or the updated function wasn't being used).
- Or the skip was removed in an attempt to implement a decoding method (since you see "Successfully decoded using HTML/API method" in the log, which suggests a new decoding attempt was added).

Double-check the `resolve_google_news_url` function in your code. According to the planned fix, it should look like this [3]:

```
def resolve_google_news_url(self, url: str) -> Optional[str]:
    if "news.google.com/rss/articles/" not in url:
```

```
            return url
        self.logger.warning(f"Skipping Google News redirect URL (causes redirect
loops): {url[:100]}...")
        return None
```

If your implementation differs (for example, if you attempted a custom decode routine instead), that could explain why the redirect URLs were still processed and led to errors. **Ensure that the skip logic is in place** if you want to avoid these failures. Skipping means those items won't be scraped at all (preventing the errors). The plan recommended this because such links are "not meant for direct scraping" [4].

Solutions and Next Steps

1. Reinstate Skipping of Google News URLs (Recommended)

Given how problematic these Google News redirects are, the simplest fix is to **skip them entirely**. This was the critical fix suggested to eliminate the redirect loop errors [2]. Make sure your code is indeed doing that. With skipping:
- The scraper will log a warning for each Google News URL and **not attempt any decoding or extraction**. e.g., `Skipping Google News redirect URL (causes redirect loops): https://news.google.com/rss/articles/CBMi...`
- Those articles will be marked as skipped (or failed gracefully) and won't count towards extraction failures. Your success rate will no longer be stuck at 0% due solely to these loops.

Trade-off: By skipping, you won't get content from Google News-sourced items. However, you've already seen that the current decode approach isn't yielding content either. It may be better to skip these to allow the pipeline to succeed on other sources (and avoid wasting time on impossible redirects). In the long run, you can incorporate more direct RSS feeds from publishers (as you started to do with `handelszeitung`, `bilanz`, etc.) so that important articles come through those channels instead of Google News.

2. Use an Updated Decoder Library (Alternative)

If skipping is not desirable (because you **must** retrieve those news articles), consider using a specialized solution for Google News URLs. Notably, there's a Python package called `googlenewsdecoder` (by SSujitX) that was created to handle the new Google News redirect format [5] [6]. This library implements the logic (converted from a TypeScript solution) to fetch the actual link.

To use it: - Install the package: `pip install googlenewsdecoder` (make sure to upgrade to the latest version).
- Use it in your `resolve_google_news_url`. For example:

```
from googlenewsdecoder import gnewsdecoder

def resolve_google_news_url(self, url: str) -> Optional[str]:
    if "news.google.com/rss/articles/" not in url:
        return url
```

```
    try:
        decoded_url = gnewsdecoder.get(url)  # hypothetical usage; refer to
package docs
        if decoded_url:
            self.logger.info(f"Decoded Google News URL to: {decoded_url}")
            return decoded_url
    except Exception as e:
        self.logger.error(f"Google News decode failed: {e}")
    # If decoding fails, skip it to avoid loops
    self.logger.warning(f"Skipping problematic Google News URL: {url}")
    return None
```

(The actual usage of `googlenewsdecoder` *may differ; consult its documentation or source. The idea is that it will handle contacting Google's hidden endpoints, such as using the* `/read` *API, to retrieve the original article link.)*

Using this library or a similar approach, you might be able to get the real article URLs. Keep in mind, however, that this introduces complexity and potential points of failure (Google could change things again or rate-limit these requests). According to a Stack Overflow answer, **the new Google News redirects might not be reliably solvable client-side** without Google's cooperation [1]. So, use with caution.

3. Alternative Workaround: Direct `news.google.com` Navigation

Another approach (if you prefer not to add a new dependency) is to **let the headless browser handle the redirect**, but with a tweak. Instead of trying to decode via requests, use Playwright to open the Google News URL as a page and let it execute whatever redirect or script is there. For example, you could modify the MCP prompt to:

- Navigate to the `news.google.com/rss/articles/...` URL directly in the browser.
- If it shows a Google page, the agent might need to click "Continue" or handle a redirect. In some cases, Google News links might immediately redirect to the source if opened in a real browser. If not, the agent could look for a meta-refresh or link in the page content.

This is tricky to script in the prompt, but you could attempt something like: *"If the page is a Google News redirect page, try to find the link to the original article or let the page redirect."* The success of this is not guaranteed – it depends on how Google serves those RSS links. Given your current log, the Playwright agent didn't get a chance to do much because it likely hit the `gstatic` endpoint and stopped. So this may not be straightforward.

4. Verify OpenAI MCP Configuration (Minor)

The repeated `Connection closed` errors from `mcp_use` might also indicate an OpenAI API or browser issue. Since you enabled the MCP+Playwright agent, ensure that:
- Your OpenAI API key is set and the model name is correct. (In code, it defaulted to `"gpt-5-mini"` which is not a real model. You might need to set `MODEL_MINI` env var to `"gpt-3.5-turbo"` or another valid model name.) If the model name was invalid, the agent could be failing to run, which might explain the immediate "Connection closed".

- The MCPAgent is properly installed and the Playwright browser can launch. The log indicates the MCP agent initialized, so that's good. The "Connection closed" could be the LLM connection, as noted. Double-check by running a simple MCPAgent query on a known good URL to see if it works.

This is secondary to the main issue, but worth checking. If the agent wasn't actually running due to an API issue, fixing the URL alone might not yield content until the LLM calls succeed.

Conclusion and Next Run Expectations

The core problem is that **Google News RSS links are not directly scrapeable** under Google's new format. The previous suggestions intended to skip them, which would avoid the errors (as confirmed in the research notes: *"Eliminates all Google News redirect loop errors"* [2]). If you haven't done so, **implement the skip logic or integrate the dedicated decoder** to handle these links.

Once you do that, rerun the pipeline. You should see:

- Fewer or no errors in the scraping step. Either the items will be skipped with warnings (no 404s or connection errors), or the decoder will output real publisher URLs that can be fetched.
- A **non-zero success rate** if some articles from other feeds are extracted. Even one successful extraction will confirm things are improving.
- The summarization step will then process those extracted articles (currently it said "No articles found to summarize" because none were extracted).

If after skipping/decoding you still get 0 extracted articles, then the issue might be elsewhere. But given the log, the Google redirect loop is the obvious showstopper. Addressing that should significantly improve the outcome.

Remember, it's known from community feedback that **we cannot reliably decode Google's new RSS URLs purely via base64 or simple means** [1]. So either rely on Google's own redirection (via a browser or API) or avoid those links. In practice, focusing on direct RSS feeds from publishers might be more stable for your use case.

By fixing the URL resolution and ensuring the MCP agent is configured properly, your pipeline should proceed past Step 3 with actual content. Good luck, and consider running a test on just a few articles to verify everything before the next full daily run!

Sources:

- Stack Overflow – Explanation of Google News URL changes (2024) [1]
- News Analysis Scraping Fixes – Google News URL skip solution [2]
- Stack Overflow – Reference to `googlenewsdecoder` library for new Google News URLs [6]

---

[1] parsing - Decoding Google News Redirect URLs - Stack Overflow
https://stackoverflow.com/questions/78791573/decoding-google-news-redirect-urls

2 3 SCRAPING_FIXES_SUMMARY.md
https://github.com/ClaudioLutz/NewsAnalysis_2.0/blob/230c2ca554e4a1a5ec017602817551531d1c7208/SCRAPING_FIXES_SUMMARY.md

4 scraper.py
https://github.com/ClaudioLutz/NewsAnalysis_2.0/blob/230c2ca554e4a1a5ec017602817551531d1c7208/news_pipeline/scraper.py

5 This script decodes Google News generated encoded, internal ...
https://gist.github.com/huksley/bc3cb046157a99cd9d1517b32f91a99e?permalink_comment_id=5150878

6 python - Decoding encoded Google News URLs - Stack Overflow
https://stackoverflow.com/questions/51131834/decoding-encoded-google-news-urls