ChatGPT

Resolving RSS Parsing and Google News Redirect Errors

RSS Feed Parsing Issues (Malformed Feeds)

Several feeds (e.g. Luzernerzeitung, Tagblatt, Aargauerzeitung, Handelszeitung, etc.) logged warnings like "not well-formed (invalid token)" and then "No entries found… skipping." This indicates the RSS XML was **malformed** – containing illegal characters or bad formatting. Your code already uses **feedparser** in "bozo-tolerant" mode, which tries to recover from XML errors [1]. Feedparser flagged a `bozo_exception` (malformed XML) and, finding no entries, skipped those feeds. In short, the feeds themselves are the issue, not your code.

Why it happens: Publishers sometimes generate invalid XML (e.g. unescaped `&` or bad encoding), causing parse errors. For example, a stray character can make the XML "not well-formed." Feedparser handled some errors but couldn't extract any entries, hence the skip.

How to fix: If these feeds are important, you have a few options:

- **Use a more forgiving parser or pre-clean the XML:** For instance, use `lxml` with recovery or BeautifulSoup to clean the feed text before parsing. Example in code:

```
import requests
from lxml import etree
response = requests.get(feed_url)
parser = etree.XMLParser(recover=True)  # ignore malformed tokens
root = etree.fromstring(response.content, parser=parser)
# then extract items from root...
```

This can parse through some errors. However, this is a bit low-level; feedparser is already doing similar recovery internally.

- **Alternate feed source:** If possible, find a correct RSS feed from those publishers. Sometimes the "arc/outboundfeeds" endpoints are buggy. The publishers might have another RSS URL or an HTML page listing articles you could scrape instead.

- **Skip or log and move on:** Since your code already logs a warning and skips when `feed.entries` is empty, it's acceptable to continue skipping these until the feed providers fix their XML. You might add logic to **notify or retry later** in case it's a transient issue.

In summary, the RSS parsing warnings are due to malformed XML from the source. Your feedparser usage is fine – it correctly identified the problem and skipped those feeds. Unless you need those sources urgently, the simplest approach is to continue skipping or use alternate sources.

Google News Redirect Links Causing 404 (KidsManagement Domain)

The more critical errors are during **Content Scraping** for Google News aggregator links. In the logs, every Google News RSS link (those `news.google.com/rss/articles/CBMi...` URLs) was "decoded" to a strange domain: `https://kidsmanagement-pa.googleapis.com`, which returned a 404 and caused the content extraction to fail. This is why you saw repeated `404` errors and "Connection closed" for almost all articles.

Cause: Google News RSS links are not direct; they require a redirect or an API call to get the final article URL. You implemented a decoding method (likely using the hidden Google News `batchexecute` API or similar) to resolve those links. However, Google has recently changed how these redirects work. When multiple Google accounts are present or certain account contexts apply, the redirect URL can be **rewritten with an account identifier** that breaks the redirect [2]. In your case, the resolved URL pointing to `kidsmanagement-pa.googleapis.com` is an internal Google domain related to **Family Link (child accounts)**. In other words, Google appended a "kids management" context to the news redirect, causing an unreachable URL for your scraper. This behavior has been observed on Android/Chrome when a user is in a supervised (child) profile – Google routes some content via `kidsmanagement-pa.googleapis.com`, which will 404 if you're not actually that service [3] [4].

Even though your scraper isn't logged in to Google, the **batchexecute trick** might be retrieving a URL with an account context (possibly a default or a dummy one). The result is that instead of the real news article URL, you get a Google API endpoint that your code can't access (hence the 404).

Solution: The most robust fix is to **avoid trying to decode Google News redirect URLs at all**, and instead gather direct article URLs from sources whenever possible. In fact, an update to your codebase reflects this as a "critical fix" – the strategy now is to **skip Google News aggregator links entirely** because they are "complex and unreliable" [5]. You can implement a similar skip in your `resolve_google_news_url` function:

```
def resolve_google_news_url(self, url: str) -> Optional[str]:
    if "news.google.com/rss/articles/" in url:
        logging.warning(f"Skipping Google News redirect URL (causes redirect
loops): {url[:100]}...")
        return None  # Skip attempting to resolve
    return url
```

And in your scraping logic, check for `None` and skip those items [6]. This change tells the pipeline not to waste time on Google's encoded links. Instead, focus on feeds that directly provide article URLs.

By skipping these problematic links, your "Scraping content" step will no longer attempt the failing decode/redirect, and you won't see the `kidsmanagement-pa.googleapis.com` errors. The downside is you lose those aggregated articles. To compensate, consider adding the original sources' RSS feeds. For example, many of the Google News links were pointing to sources like **AP News, Finanz und Wirtschaft, Blick**, etc. You could add those sources' RSS feeds directly to your `feeds.yaml` so you get the content without Google's intermediary.

Alternative (advanced): If you *must* resolve Google News URLs via code, you'll need to update your approach to mimic a single-account, adult Google News environment. One way is using a real headless browser to follow the redirect. For instance, using Playwright directly (without the LLM agent) to navigate to the `news.google.com/rss/articles/...` URL might get the proper redirect to the article. Another approach is adjusting the `batchexecute` payload. (Google may have changed the JSON structure – the snippet that removes the last 6 elements `obj[:-6] + obj[-2:]` might need adjustment if the structure changed.) However, these methods are brittle, as Google doesn't officially support this and could change it anytime. The **recommended approach** is to skip these and use official feeds.

In summary, to fix the errors:

- **Skip Google News RSS links** in your scraper. Modify `resolve_google_news_url` to detect Google's encoded URLs and return `None` or otherwise mark them to skip. In the scraping loop, if `resolved_url` is None, log and continue to the next article. This prevents the endless 404 failures [7]. Your updated code might look like:

```
resolved_url = self.resolve_google_news_url(item_url)
if not resolved_url:
    logger.warning(f"Skipping problematic redirect URL: {item_url}")
    continue  # skip to next article
# else proceed with trafilatura or MCP on resolved_url
```

- **Add direct feeds** for important sources to reduce reliance on Google News. This way you get the same articles without the redirect hassle.

By making these changes, your pipeline should run without those errors. You'll no longer get stuck on Google's redirect (no more `kidsmanagement-pa.googleapis.com` 404s), and the success rate of content scraping will improve dramatically. The official solution in your codebase was to avoid these redirects entirely, which aligns with best practice given Google's unpredictable redirect behavior [5].

---

[1] collector.py
https://github.com/ClaudioLutz/NewsAnalysis_2.0/blob/230c2ca554e4a1a5ec017602817551531d1c7208/news_pipeline/collector.py

[2] Google News, Discover links showing 404 Not Found? Here's how to fix
https://www.bleepingcomputer.com/news/technology/google-news-discover-links-showing-404-not-found-heres-how-to-fix/

[3] [4] Disable Google Chrome Kids Management API : r/chrome
https://www.reddit.com/r/chrome/comments/1g8hsbu/disable_google_chrome_kids_management_api/

5 6 7 scraper.py

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