Reading One Conversation Out of an AI Export: RECLAiM Against the Raw File

Author: AURiX (OnwardAI) **Date of the run:** 2026-11-06, 11:09-11:10 UTC, on the owner's own machine **Bench:** PAPERS_2026-11-06_bench\bench_2_reclaim_vs_raw.py; raw output result_2_20261106T111009Z.json, log_2.txt; tables SUMMARY_TABLES.md

Abstract

A ChatGPT export arrives as three JSON files, 201,133,174 bytes in all, holding 257 conversations. Each conversation is a tree. The quick way to read it walks the first branch at every fork (children[0]), and that walk reached 28,731 of the 57,505 message nodes, or 50.0%. RECLAiM 2.7.0 turned the same export into one readable text file per conversation in 53.9 seconds, with 50,096 messages. After that one run, finding and reading a named conversation took 0.0002 to 0.0028 seconds and opened 72 to 864 KB of plain text. The raw route took 1.20 to 1.52 seconds and opened all 201 MB of JSON. RECLAiM loses one case: a single one-off lookup is faster in the raw JSON than running RECLAiM first.

The job

Someone wants to find one conversation in their exported AI history and read it the way it happened, in order and complete.

The go-to, and why a seat reaches for it

A seat (an AI assistant working on the files) opens conversations.json with a JSON library, looks for the title, and walks the messages. It does this because the file is right there and the code is ten lines. The trap is the tree. ChatGPT stores every regenerated answer and every edited question as a branch. Code that follows the first child at each step reads one path through the tree and silently skips the rest. reClaim versions 2.1 to 2.4 made exactly this mistake. THE_BATON's RECORD section records the fix in v2.5.0: the full tree held 257 conversations and 49,677 messages, against roughly half that before.

The instrument

RECLAiM (AURiX\current\reclaim_v2.py, reporting itself as v2.7.0 tonight) reads the whole export once. It writes one .txt and one .json per conversation, each opening with a header that gives the title, the original dates, the message count and the fingerprint of the source. It runs on the owner's machine with no network.

Method

1. Source: the owner's own export, Desktop_ARCHIVE_Archives_Exports\2026-04-25_openai_export_final.zip (1,288,701,260 bytes). Its three conversations-00N.json files were copied into a scratch folder, /tmp/bench2_export.

2. RECLAiM was run once on that folder and timed: python3 reclaim_v2.py /tmp/bench2_export --output /tmp/bench2_reclaim_out --platform chatgpt --no-attachments.
3. Whole export: messages were counted by three walks of the raw tree using one rule (a node counts if it carries text or non-text content). The walks were children[0] from the root, the branch the user last saw (current_node back to the root), and every node. They were set against the message counts in RECLAiM's file headers.
4. Five target conversations: the three where children[0] loses the most, the first conversation, and the conversation of median size. For each, the raw route (load all three files, find the title, walk the tree) and the RECLAiM route (scan the first line of each .txt for the title, read that one file) were timed, and the bytes each route opens were counted.

Results

Whole export (257 conversations):

walk
raw tree, children[0] only
raw tree, the branch last seen
raw tree, every node with content
RECLAiM, sum of file headers

children[0] loses messages in 70 of the 257 conversations.

Five lookups:

conversation	raw: find + read (s)	raw: bytes opened	raw children[0] msgs	raw tree nodes	RECLAiM: find + read (s)	RECLAiM: bytes opened	RECLAiM msgs
A (largest loss)	1.52	201,133,174	464	1,705	0.0020	846,246	1,610
B	1.40	201,133,174	50	1,253	0.0023	742,607	1,030
C	1.51	201,133,174	81	1,196	0.0007	864,402	1,142
D (the first conversation)	1.20	201,133,174	8	129	0.0002	94,973	115
E (median size)	1.37	201,133,174	76	76	0.0028	71,938	53

The raw file is not something a person can read. conversations-000.json has 1,060,785 lines, and its longest line is 83,949 characters.

Where the instrument loses

- **A single lookup.** RECLAiM's one-time conversion took 53.9 seconds, and loading the raw JSON took 1.25 seconds. For one lookup and nothing after, the raw route is faster. RECLAiM pays for itself after about 39 lookups (53.9 / 1.4).
- **Not every node is written as a message.** RECLAiM's 50,096 is below the 57,505 tree nodes counted by the loose rule. The gap includes ChatGPT's hidden reasoning ("thoughts" 1,490 nodes, "reasoning_recap" 822) and browsing and tool displays. Conversation E shows it: 76 nodes in the tree, 53 messages in RECLAiM's file, with no branches lost. Whether every one of those nodes belongs in a reader's copy was **not measured** tonight.
- **The in-browser portal** (reclaim-portal\ on the site) was **not measured**. It needs a browser run.

How to rerun

1. Extract conversations-000.json, -001 and -002 from the export zip into /tmp/bench2_export (any empty folder; change EXP in the script to match).
2. python3 PAPERS_2026-11-06/_bench/bench_2_reclaim_vs_raw.py
3. python3 PAPERS_2026-11-06/_bench/summarize.py rebuilds the tables.

Sources

- SELFiX\PAPERS_2026-11-06_bench\result_2_20261106T111009Z.json, log_2.txt (run 2026-11-06 11:09-11:10 UTC)
- SELFiX\AURiX\current\reclaim_v2.py (v2.7.0 as reported by the run)
- Desktop_ARCHIVE_Archives_Exports\2026-04-25_openai_export_final.zip
- SELFiX\THE_BATON.md, RECORD section (the children[0] bug in v2.1-v2.4; 257 conversations / 49,677 messages under v2.5, recounted 2026-07-10)
- SELFiX\AURiX\reclaim_output\RECEIPT_chatgpt_export_2026-04-13_run_20261009T1124Z.txt (an earlier export: 255 conversations, 49,610 messages, every one matched word for word)