# Coding QuIP data with AI and MapCat

31 Aug 2026

A QuIP corpus is the kind of data AI coding was built for: many interviews, each dense with statements about what led to what, all needing the same treatment. Hand-coding sixty transcripts takes weeks and the coder gets tired somewhere around transcript forty. The app will do the same pass in an afternoon and attach a quote to every link it produces, which means you can check its work rather than trust it.

This page covers what changes for a QuIP analyst. The general account is the coding workflow, and the control-by-control reference is the AI coding panel.

## How the app codes

The app codes claim by claim rather than asking for one connected story per interview. You get every link in the text, at the cost of a map that arrives in fragments and needs rejoining. That suits QuIP, where exhaustive coverage of what respondents said matters more than a tidy narrative, and where a missed claim is a missed voice.

Chunk size turns out to matter more than prompt wording. Models satisfice: hand one a long transcript and it reports a plausible handful of claims and stops, whatever you told it. In our experiments, chunks of 2,000 characters found half as many links again as chunks of 4,000 (the experiments are synthesised here). Small chunks bring their own problem, because a cause mentioned on page one and its effect mentioned on page six never meet, so the app counters with segment accounting and a join-islands pass at the end.

For a large study, code a sample first. With sixty interviews, code ten, read what came back, work out exactly why any output is wrong or thin, change the instruction and run again. Then scale up. Make the sample random or stratified by the groups you care about, so you do not tune the instruction to one untypical district.

## Where MapCat comes in

MapCat is the chat assistant in the bottom-right corner of the app. It is available on every account, including accounts with no AI subscription, because by default it does not see your project data at all. In that default state it answers questions about the app and about causal mapping while you work.

Turning on **Edit my data with AI** is your consent to send this project's data to the model, and it changes what MapCat can do. It can then read your factors, sources and links, run the coding for you after

asking a couple of plain questions about how you want it done, build a report deck, save a view as a bookmark and reopen it later to adjust. Every change asks you to confirm before anything happens to the project.

For a QuIP analyst the useful part is that MapCat holds your aim across the conversation. State at the start what the evaluation needs to find out, and the set-up questions, the coding conventions it proposes and the report it builds all answer to that aim rather than to whatever you last typed. It reports the conventions it chose in a summary before the run, so the decisions stay yours even when you did not make them one at a time.

## What does not change

The coding model finds and records claims. Deciding what those claims are worth is still the evaluator's job, and in a QuIP study it is the job the method exists to discipline. AI coding makes the evidence base bigger and traceable; it does not make the causal inference for you, and QuIP's own supports for getting from claims to conclusions apply exactly as before.

Quality assurance matters more, not less, when a machine did the coding. Quality assurance at each step works through where to check: whether the coder found the claims that are there, whether it invented any that are not, whether the factor labels hold together, and whether a link's quote really supports it. Read a sample of quotes against their links before you report anything.

---

*I sometimes use AI to help draft these pages, but the final text is ours.*