AI coding overview

2 May 2026

Summary

This page is an overview of the choices you face when doing AI coding (qualitative causal coding with AI) inside the Causal Map app.

- For step-by-step app docs see AI coding
- For the AI Answers panel (asking questions of coded or uncoded text) see AI answers panel
- For the broader argument about why we use AI as a low-level coder rather than as a black box, see AI in evaluation actually show your working! and Just add rigour Three do's and don'ts.

Workflow

Someone arrives with a stack of documents and asks: can you code these? It depends first on various parallel setup decisions. After coding, you face a separate decision about how, or whether, to recode the labels.

Decisions before coding

Codebook strategy: how free?

You can start with a full codebook, a partial codebook, or nothing.

A *forced* codebook restricts the model to your labels (for example from a theory of change). If a causal claim mentions a factor not in the codebook, no link is coded.

Most often you provide a codebook but allow the model to invent labels when nothing fits. This is harder to manage: telling the model it *can* improvise seems to confuse it a bit, and codebook coverage drops.

Four codebook strategies:

1. Stick only to the codebook. Anything that doesn't match is dropped.
2. Stick to the codebook mostly, but let the AI code other things too. Tell it to flag the new ones with a tag like `[new]` or a trailing `*`, so you can find and review them later.
3. Compromise. Use only top-level labels from the codebook, but let the AI improvise the second part of a hierarchical label. See Hierarchical coding.]
4. Free coding

Related to the above...

Label style: In vivo or abstract

When free coding, you can ask for *in vivo* labels (close to the original text) or for more abstract labels. There are many ways to instruct: "talk like a social scientist" or "talk like a local newspaper editor".

Label style: Using tags

There's often more to a coding task than just labels. *Tags* are short bits of text in brackets attached to labels: `stressed patients (before surgery)`, `stressed patients (after surgery)`. Tags help you build up a system of labels from smaller parts.

Label style: Opposites

See !Opposites and sentiment in AI coding and Opposites.

Label style: Hierarchical

You can impose hierarchical labels even when free coding. See Hierarchical coding.

Columns

We also often ask for *columns* unrelated to the labels, such as a sentiment assessment. Columns can be useful for any systematic attribute you can code consistently across most links. Note the difference: columns code attributes of *links*; tags code attributes of *factors*.

Asking the model to fill extra columns alongside coding is extra work for it. Expect either coverage (links found) or precision to drop.

Columns: Sentiment

Zero-codebook coding usually leads on to magnetic soft recoding, and in embedding space `decrease in X` sits close to `increase in X`.

With an explicit codebook, you can get round this by (maybe) suggesting both sides of any factor likely to appear positively and negatively, e.g. `increased income` and `decreased income`.

When you don't specify a codebook, get the model to code sentiment for each link.

Sentiment is a kind of column.

A sentiment column lets you tell them apart.

Sentiment can also be interesting for other reasons.

Other things which are important in the prompt

Often you want to tell the model what the work is for: the context, named entities, the audience. We *iterate* on the prompt, trying versions and comparing results.

Coding style: holistic or claim-by-claim?

Two main approaches.

Holistic: we ask the model for a Mermaid diagram of the causal network, then convert it into a links table. The model decides what the main links are, but we still ask for a quote behind each one. This may handle longer sources better, but we don't really know.

Claim by claim: we ask the model to find each individual causal link. Hundreds of tweaks and heuristics later, this style still struggles to tell a connected story even when the text contains one. Suppose the text supports A to B to C to D, but the model lazily codes A to B and C to D, using slightly different labels for B and C. Claim-by-claim coding only really works if you plan to recode afterwards: you hope a later recoding pass spots that B and C mean the same thing and rejoins the chain.

Model

For relatively straightforward cases, newer or bigger models are not necessarily better. We have had good results with Gemini Flash, for example.

How many iterations?

Additional iterations can be useful:

- For checking and improving quality
- For increasing coverage (finding more actual causal claims)
- For adding additional information (e.g. more columns)

Multi-stage prompts (separated by `====`) let you split coding into sequential passes. The UI does this for you. Mechanics in !1010-Auto-coding and AI coding under "Prompt sections".

Chunk and sample strategy

If a single source is short, you can map the whole thing in one go. More often you have either much longer sources, which the app breaks into chunks for you, or many sources.

Recall and precision

A pretty hard rule: the more text you give the model at once, the less dense its coding gets. One page might yield 20 links; 5 pages might also yield 20 links. Sometimes the 20 it picks out of 5 pages really are the most important, but this is not certain, and you are leaving the model to decide. Often you just want better recall, and that means smaller chunks. Don't give the model too much freedom to decide what counts as important.

What does it select

Bigger chunks mean sparser coding. There is a "code everything" setting that does not each source at all: the effect depends on how long your sources are. Otherwise, work in smaller chunks. Aiming to pick up *every* causal claim is unrealistic, but the smaller the chunk, the more you'll catch.

Iterative coding strategy with bigger corpora

If you have more than around 100 pages, work on a sample first. The app has features to take a random sample of sources, or a stratified random sample within source groups. The AI coding panel also offers a sample-only run (see AI coding). Try the sample, review, adjust your strategy, perhaps update the codebook, then run a larger sample. With 1000 pages you might code 100, adjust, code another 300 (deleting the first attempt), and if that looks good, finish with the remaining 700.

Recoding style

Once the first coding run exists, the next question is how to deal with overlapping labels. See Coding and recoding — Dealing with overlapping labels.

Recoding from scratch means arriving at a clean revised codebook and then recoding everything from the beginning. That's more expensive and slower, but usually gives better results than magnetically recoded labels alone (see Coding and recoding — Dealing with overlapping labels).

- *Hard recoding*: revise the codebook and code again.
- *Links recoding*: use AI Answers / Links to recode links, with quote and context available.
- *Factors recoding*: use AI Answers / Factors to recode factor labels directly.
- *Soft recoding*: use clustering or magnetic labels as a softer recoding layer.

Related

- chapter intro
- AI coding: app docs for the AI Coding panel
- AI answers panel: app docs for AI Answers
- !1010- Auto-coding: older detailed notes on the auto-coding prompt
- !Opposites and sentiment in AI coding
- Coding and recoding — Dealing with overlapping labels
- ! Autoclustering with embeddings
-
- AI in evaluation actually show your working!: the transparency argument
- Just add rigour Three do's and don'ts: do's and don'ts for AI text analysis
- Causal mapping is easy to automate transparently, so is a great fit for scaling with AI