

CHAPTER

Working Papers

CONTENTS

[Conversational AI -- Analysing Central Bank speeches](#)

[Rigour and causal pathways](#)

[A simple measure of the goodness of fit of a causal theory to a text corpus](#)

[A simple measure of the goodness of fit of a causal theory to a text corpus](#)

[Minimalist coding for causal mapping](#)

[Magnetisation](#)

[A simple measure of the goodness of fit of a causal theory to a text corpus](#)

[Causal mapping as causal QDA](#)

[Rigour and causal pathways](#)

[Rigour and causal pathways](#)

Conversational AI -- Analysing Central Bank speeches

Working papers hub (project overview)

This folder is a set of working papers about **causal mapping as qualitative evidence management**: we code *reported causal influence claims* from text as a links table with provenance, then analyse the resulting evidence base through explicit transforms and queries.

The overall aim is to keep the core representation **minimal and auditable**, while still supporting powerful downstream analysis (filter pipelines, standardisation/recoding, coverage/fit diagnostics), including workflows that use LLMs as low-level assistants for extraction and labelling.

Core papers (start here)


- [Minimalist coding for causal mapping](#): the core coding stance (“barefoot” link coding), why it is useful, and where it breaks.
- [A formalisation of causal mapping](#): companion spec—data structures + conservative rules for aggregation/query.
- [Causal mapping as causal QDA](#): positioning for qualitative methods / CAQDAS audiences.

Practical extensions (operations on a links table)

- [Magnetisation](#): soft recoding with “magnets” (standardise labels at scale without re-coding quotes).
- [A simple measure of the goodness of fit of a causal theory to a text corpus](#): coverage-style diagnostics for ToC fit.
- [Combining opposites, sentiment and despite-claims](#): opposites transforms, sentiment as an annotation layer, and “despite” link typing.
- [Hierarchical coding](#): hierarchical labels (:) and zoom-style simplification.

Related notes / fragments / examples

- [!!!Qualitative Split-Apply-Combine](#): small-Q framing; causal mapping as a SAC variant; where genAI fits.
- [250! causal mapping turns QDA on its head](#): a short argument/fragment (kept for reuse).

- 
- [Conversational AI -- Analysing Central Bank speeches](#): worked example of “clerk vs architect” (auto-extraction + magnet-style structuring).

Rigour and causal pathways

Abstract

This paper explains our **Minimalist / Barefoot** approach to coding causal claims in text as simple directed links (“X influenced Y”), developed through large-scale practical coding and now implemented in the Causal Map app. We write it now because, although our previous work motivates causal mapping in evaluation (Powell et al., 2024), shows how QuIP-style “stories of change” elicit natively causal narrative evidence (Copestake et al., 2019), demonstrates ToC validation by comparing empirical maps with programme theory (Powell et al., 2023), and shows that genAI can extract links exhaustively with quotes as a low-level assistant (Powell & Cabral, 2025); (Powell et al., 2025), none of these papers is a standalone, reader-facing account of **the coding stance itself**.

Intended audience: evaluators / applied qualitative researchers who want a teachable causal coding protocol, and AI/NLP readers who want a simple, auditable target representation of causal content in text.

The contribution here is therefore to make the minimalist coding commitments explicit, teachable, and citable: what exactly counts as a coded causal claim, what we deliberately do *not* encode (effect size, causal typing, polarity-by-default), how evidence/provenance is kept auditable via quotes, and the limits of the approach (e.g. blockers/enablers and conjunctions). A companion formalisation paper [A formalisation of causal mapping](#) makes key parts of the method more precise (data structures, constraints, and conservative inference rules), but the focus here is the narrative rationale and practical coding guidance.

Unique contribution (what this paper adds):

- A definition of the **minimalist/barefoot** coding stance (“X influenced Y, with provenance”) and what it deliberately does *not* encode.
- An account of why this stance is useful at scale (including AI-assisted extraction) and where it fails (enablers/blockers; conjunctions/packages).
- Examples of additional extensions / transforms, called “filters” in the Causal Map app. Both the initial definition of a links table and each additional filter is described in terms of syntax and of semantics (interpretation rules).

As before, we treat causal mapping as three tasks

- Task 1, data gathering, not covered here;
- Task 2, coding: creating **evidence-with-provenance** (a links table)
- Task 3, analysis: a sequence of transforms of the original links table: the final inte

Project context (how this paper fits)

This paper is part of a small set of new working papers

- Companion formal spec: [A formalisation of causal mapping](#)
- QDA positioning: [Causal mapping as causal QDA](#)
- Practical transforms/diagnostics: [Magnetisation](#); [A simple measure of the goodness of fit of a causal theory to a text corpus](#); [Combining opposites, sentiment and despite-claims](#); other transforms to come later
- A worked “AI clerk vs human architect” example: [Conversational AI -- Analysing Central Bank speeches](#)

Introduction

I'll start by describing the **"minimalist" approach** to coding causal statements used for QuIP and developed originally by James, Fiona and colleagues at BathSDR and developed and further formalised at Causal Map Ltd in collaboration with BathSDR. This formalisation lives inside the [Causal Map app](#). Then we will show how we can extend this approach with useful transformations. Finally I will try to answer the question of whether it can help us deal with more complicated constructions like enabling and blocking and whether this could help us with mid-range theory. As an appendix I'll add a more detailed overview of minimalist causal coding.

The minimalist approach is notable because it is based in our **joint experience of coding thousands and thousands of stakeholder interviews and other data such as project reports**, mostly from international development and related sectors, as well as coding hundreds of thousands of pages with AI-assisted coding. These have nearly always involved **multiple sources talking about at least partially overlapping subject matter**. So this coding produces individual causal maps for each source, which can then be combined in various ways -- rather than constructing single-source maps of expert thinking (Axelrod, 1976) or the collective construction of a consensus map (Barbrook-Johnson & Penn, 2022).

This paper sits alongside (and builds on) four related contributions. First, our evaluator-facing account argues that causal mapping is best treated primarily as a way to assemble and organise **evidence-with-provenance**, keeping the subsequent evaluative judgement about “what is really happening” distinct (Powell et al., 2024). We adopt that stance here: a coded link is first and foremost “there is evidence that a source claims X influenced Y”, not a system model with weights or effect sizes. Second, QuIP-style evaluation practice shows how “stories of change” can be elicited in a goal-free / blindfolded style to reduce confirmation bias, yielding narrative data that is natively causal (change plus reasons) and therefore well-suited to parsimonious link coding (Copestake et al., 2019). Third, our ToC comparison case study shows how empirical causal maps can be used as a disciplined way to check a programme’s Theory of Change against beneficiaries’ narratives, and to support evidence-based adjustment of “middle-level theory” rather than just project-level reporting (Powell et al., 2023). Fourth, our AI-assisted causal mapping work shows that this minimalist stance is also a practical sweet spot for automation: we can use genAI as a low-level assistant -- because the minimalist coding task is so relatively easy -- while keeping human judgement focused on the few high-leverage decisions (prompt design, verification, and synthesis choices) (Powell & Cabral, 2025); (Powell et al., 2025). The present paper extracts and

clarifies the core “minimalist” coding commitments that make that workflow workable and checkable.

Our experience has been that the vast majority of causal claims in these kinds of texts are easily and satisfactorily coded in the simplest possible form "X causally influenced Y". Explicit invocation of concepts like enabling/blocking, or necessary and/or sufficient conditions, or linear or even non-linear functions, or packages of causes, or even the strength of a link, are relatively rare. The causes and effects are not conceived of as variables, the causal link is undifferentiated, without even polarity, and if any counterfactual is implied it remains very unclear.

This approach is what we call **"Minimalist" or "Barefoot" Coding**.

Why minimalist coding?

Using more sophisticated, non-minimalist coding such as DAGs or fuzzy cognitive maps or whatever allows one to code linear or even non-linear causal influences of single or even multiple causes on their effects. One can do the "coding" by simply writing down (using appropriate special syntax) the connections, because one is an expert, and/or one can verify such statements statistically on the basis of observational data. Thus armed, one can make predictions or have sophisticated arguments about counterfactuals. But using minimalist coding we cannot do that, because our claims are formally weaker and therefore our inference rules are weaker. What we *can* do is still really interesting. We can ask and answer many different kinds of useful questions like:

- what are the main influences on (or effects of) a particular factor, according to the sources?
- what are the upstream, indirect influences on (or effects of) a particular factor, bearing in mind [The transitivity trap](#)?
- how well is a given programme theory validated by the respondents' narratives? (We can do this basically by using embeddings to get measures of semantic similarity between labels and aggregate these as a goodness of fit of theory to data.)

That is all exciting and useful. It's a surprisingly simple way to make a lot of sense out of a lot of texts which is, with caveats, almost completely automatable.

Minimalist coding principles

The 90% rule

We have found that it is pretty easy to agree how to apply minimalist coding to say 90% of explicit causal claims in texts, without missing out essential causal information, whereas it is very difficult to find appropriate frameworks to cope with the remaining 10%.

Fewest assumptions

Minimalist coding is perhaps the most primitive possible form of causal coding which makes no assumptions about the ontological form of the causal factors involved (the "causes" and "effects") or about *how* causes influence effects. In particular we do not have to decide if cause and/or effect

is perhaps Boolean or ordinal, or if perhaps multiple causes belong in some kind of package or if there is some kind of specific functional relationship between causes and effects.

An act of causal coding is simply adding a link to a database or list of links: a link consists of **one new or reused cause label and one new or reused effect label**, together with the highlighted quote and the ID of the source.

A statement (S) from source Steve:

I drank a lot and so I got super happy

can be trivially coded minimalist-style as

I drank a lot --> I got super happy (Source ID: Steve; Quote: I drank a lot and so I got super happy)

That's it.

Causal maps

Crucially, we can then display the coded claims for individuals as a graphical causal map, and we can also display the entire map for all individuals and/or maps filtered in different ways to answer different questions. There is a handful of other applications (Ackermann et al., 1996) (Laukkanen, 2012) for causal mapping which also do this; but as far as we know, only Causal Map also allows direct QDA-style causal coding of texts.

Data structure

Although we have the option of creating additional tags associated with each link (where many approaches would for example code the polarity of a link) this is not central to our approach.

We don't use a separate native table for factor labels: they are simply derived on the fly from whatever labels happen to be used in the current table of links. This makes data processing simpler and also suggests an ontological stance: causal factors only exist in virtue of being part of causal claims.

We do however have an additional table for source metadata including the IDs of sources, which can be joined to the links table in order, for example, to be able to say "show me all the claims made by women".

Causal powers

When a source uses causal language, we treat it as a claim that one thing made a difference to another (rather than a mere temporal sequence) *in virtue of its causal power to do so*.

Causal influence, not determination

We believe that it's rare for people to make claims about causal determination: someone can say that the heavy drinking made them super happy and then also agree that the music had a lot to do with it too, without this feeling like a contradiction.

Not even polarity

We differ even from most other approaches which are explicitly called "causal mapping" in that we do not even "out of the box" record polarity of links (to do so would involve making assumptions about the nature of the "variables" at each end of the link as well the function from one to the other).

The Focus on Cognition

In the minimalist approach, **we are quite clear that what we are trying to code is the speaker's surface cognitions and causal thinking**, while the actual reality of the things themselves is simply bracketed off at this stage, either to be revisited later (because we are indeed interested in the facts beyond the claims) or not (because we are anyway interested in the cognitions).

Staying on the surface

At Causal Map, we rarely make any effort to get beneath the surface, to try to infer hidden or implicit meanings. This is particularly well-suited to coding at scale and/or with AI. Our colleagues at BathSDR do this a bit differently, spending more effort to read across an entire source to work out what the source *really* meant to say.

Closer to the cognitive truth

It's really easy to code statements like (S) using minimalist coding. The trouble on the other hand with trying to use more sophisticated frameworks is that they are nearly always *ontologically under-determined*. For example, even a simple approach like Causal Loop Diagramming is function-based and requires at least a monotonic relationship between the variables: something like, *the more* I drink, *the happier* I get (in addition to which we have to code the actual, factual claims: I did drink a lot, and: I did get super happy). But is that what the speaker meant? How do we know if the speaker has say a continuous or Boolean model of "drinking"? If Boolean, what is the opposite of drinking a lot? Drinking only a little? If continuous, how do we know what kind of function they use in their own internal model?

We think it is often over-specified (and often psychologically implausible) to treat ordinary-language causal claims as if they asserted an explicit functional relationship between well-defined variables. Trying to force that kind of structure on everything turns the "easy 90%" of coding into a harder and more arbitrary task. Of course, one can decide to use a particular non-minimalist representation for a particular modelling purpose; our claim is only that this is usually not a faithful representation of what most speakers actually said or implied. For example, if we code "I got really tired because I have Long Covid", we could perhaps treat both endpoints as Boolean

variables, but what about "I got really tired because it was really hot", and "I got really tired because it was really cold"—how are we going to represent “temperature” as a single variable while preserving the speaker’s intended meaning? If what we want to do is model a system, we can pick a solution. But if we want to model *cognition as expressed in text*, many “variable semantics” choices are over-committed.

Unclear counterfactuals

More formal, non-minimalist coding has clear counterfactuals. These may often be Boolean or continuous (the volume depends on the position of the volume dial; it's a 10, so the volume is maximum, if it had been at 5, the volume would have been about half as loud, and so on). Minimalist coding arguably implies some kind of naked counterfactual, but it is not always clear exactly *what*.

General versus specific

Minimalist coding focuses primarily on **factual causal claims** which also warrant the inference that both X and Y actually happened / were the case.

Most causal claims in the kinds of texts we have dealt with (interviews and published or internal reports in international development and some other sectors) are factual, about the present or past. Sometimes we see general claims, and we often just code these willy-nilly. In any case, the distinction between general claims and claims about specific events that actually happen is often fractal (a general claim can seem specific in a broader context) and difficult to maintain completely when modelling ordinary language.

We don't code absences

Minimalist coding may be reasonably also called **Qualitative Causal Coding** or **Causal QDA coding**. It shares characteristics with some forms of coding within Qualitative Data Analysis (QDA), in particular demonstrating an asymmetry between presence and absence (a specific tag not being applied is not the same as the application of an "oppositely-poled" tag).

We do not code absences unless they are specified within the text (e.g. perhaps "because of the barking dog, the owner did not come out of the house").

While codes may be counted, the concept of a *proportion* of codes is challenging because the denominator is often unclear.

If families are talking about reasons for family disputes, and family F mentions social media use, and family G mentions homework, we do not usually interpret this as meaning that family F does *not* think that homework can also be a cause of family disputes. (This should derive formally from the interpretation of multiple causal claims).

The labels do all the work

At Causal Map Ltd, our canonical methodology initially involves in vivo coding, using the actual words in the text as factor labels. This initial process generates hundreds of overlapping factor

labels. This part is really easy (and is easy to automate with AI). Obviously, hundreds (or hundreds of thousands) of overlapping factor labels are not very useful, so we need to somehow consolidate them. Arguably, minimalist coding makes the initial coding easy but it just defers some of the challenges to the recoding phase.

!!Something about tags

Coping with many labels

We can:

- Use human- or AI-powered clustering techniques to consolidate the codes according to some theory or iteratively according to some developing theory
- Use AI-powered clustering techniques to consolidate the codes according to automated, numerical encoding of their meanings
- "Hard-recode" the entire dataset using a newly agreed codebook (see above)
- "Soft-recode" the dataset on the fly using embeddings to recode raw labels into those codebook labels to which they are most similar

Evidence strength is not causal effect size

Counts (how many times a link is coded; how many sources mention it) measure *evidence volume/breadth in the corpus*, not causal magnitude in the world. This matters because the outputs can look like a quantitative system model even when they are not one.

In the app we routinely distinguish:

- **Citation count:** how many coded mentions support a link (volume).
- **Source count:** how many distinct sources mention it (breadth / rough consensus).

These are useful for prioritising what to look at, or for filtering (e.g. keeping only links with `min_source_count = 2`), but they do not tell you whether a causal influence is “stronger” in any effect-size sense.

The transitivity trap and “thread tracing”

It is usually invalid to infer a long causal chain by stitching together links from different sources. A conservative rule is: only treat an indirect pathway ($A \rightarrow B \rightarrow C$) as supported when the *same source* provides each step (“thread tracing”), unless contexts are explicitly aligned.

The canonical failure mode is: source 1 says $A \rightarrow B$, source 2 says $B \rightarrow C$, and we mistakenly conclude that anyone told a coherent story $A \rightarrow B \rightarrow C$. In practice we handle this by “thread tracing”: for a given query we loop through sources one at a time, construct the valid paths *within each source*, and then combine only the edges that appear in valid within-source paths.

The filter pipeline as a mental model for analysis

Most analyses are built by applying a sequence of simple operations (for example: subset links to retain only specific sources; trace paths; rewrite labels; bundle duplicates; then show a map/table). Thinking in terms of a pipeline helps make the meaning of an output explicit: it is always “the result of these filters, in this order”.

Concretely, you can think of an analysis as “a links table passed through transforms”, where `|>` is a “pipe” symbol which just passes a result from the left to a new transform on the right.

e.g.:

```
Links |> filter_sources(...) |> trace_paths(...) |> transform_labels(zoom) |>
combine_opposites |> bundle_links |> map_view
```

This is also why “the same dataset” can yield very different-looking maps without any contradiction: you are simply looking at different derived views defined by different pipelines.

Positioning within qualitative research (and likely critiques)

This paper is about a **coding stance** and a corresponding **intermediate representation** (a links table with provenance), not a claim that “coding = analysis”. In standard QDA terms, minimalist causal coding is best understood as a disciplined way to build an auditable evidence base that can later be interpreted, queried, and written up (Miles, Huberman, & Saldaña, 2014; Saldaña, 2021). It is, of course, not the only kind of way to do QDA.

To reduce avoidable points of attack from mainstream qualitative social science readers, we make four explicit commitments up front:

1. **We code claims, not causal truth.** A coded link is evidence that a source *claimed* an influence relation. It is not (by itself) a causal inference claim about the world. This keeps the method compatible with both realist and constructivist sensibilities (Lincoln & Guba, 1985; Charmaz, 2014).
2. **We do not treat “counts” as effect sizes.** Counting supports prioritisation and transparency, but it is not a substitute for interpretation; frequency/breadth in a corpus is not magnitude in the world. (This paper makes that distinction explicit below.)
3. **We trade interpretive depth for auditability and scale, on purpose.** We stay close to surface causal language and preserve provenance (quotes + source ids) so that readers can check what is being claimed. This is a pragmatic stance when working with many sources and/or AI assistance; it does not deny that richer interpretive work can be valuable.
4. **Minimalist coding is not a full QDA workflow.** It is one layer that can sit alongside thematic analysis or qualitative content analysis when those are needed (Braun & Clarke, 2006; Mayring, 2000).

How a qualitative-methods reviewer might criticise this (and our response)

Below are common criticisms (often reasonable) and the corresponding guardrails/defences built into the minimalist stance.

Critique 1: “This is reductionist / strips context / decontextualises quotes”

Response: We keep provenance as first-class data (source id + quote) and treat every map/table view as a derived view of that evidence. Context is handled explicitly by joining source metadata (e.g. subgroup, time) and filtering the links table; it is not “assumed away”. When deeper within-case interpretation matters, the same links table can be re-read within full-text context, and the write-up remains responsible for integrating that context (Miles, Huberman, & Saldaña, 2014).

Critique 2: “You are smuggling in a positivist epistemology (or a realist metaphysics)”

Response: The method’s core object is **reported causal thinking** (surface causal claims), not an ontic causal model. We therefore keep the epistemic claim modest: “this is what sources said, with quotes”, and we separate that from subsequent judgement about what is happening. That separation is consistent with standard qualitative criteria emphasising transparency, audit trails, and reflexive interpretation (Lincoln & Guba, 1985; Charmaz, 2014).

Critique 3: “Coding is not analysis; links don’t ‘explain’ anything”

Response: Agreed. Minimalist causal coding produces a structured evidence base that supports multiple analyses (filters, comparisons, pathway queries) and supports more disciplined writing, but it does not replace interpretation. In practice, it can reduce time spent on low-level organisation of text so that more time can go into interpretation and argument (Saldaña, 2021; Gläser & Laudel, 2013).

Critique 4: “You are privileging causal language and missing other kinds of meaning”

Response: Yes: by design. Minimalist coding is for projects where the core questions are themselves causal (mechanisms, pathways, theories of change). When non-causal meaning is central (identity work, norms, metaphors), other QDA approaches could lead; minimalist coding can still be a useful *additional* layer rather than the whole analysis (Braun & Clarke, 2006).

Critique 5: “Automatable coding invites false precision and overclaiming”

Response: We explicitly constrain what automation is allowed to do: extract candidate links with quotes; keep everything auditable; and avoid treating the resulting network as a system model with polarity/weights “by default”. The transparency of provenance is the main defence against black-box “AI did the analysis” claims.

Extensions

An extension in general consists of syntax rules for how to construct and carry out a transformation, and semantic rules for what this transformation means.

There are many extensions one can add on top of minimalist coding. Small extensions can just add convenient functionality; other extensions can upgrade the system to other more fully-fledged kinds of coding like FCM.

This paper will only cover **hierarchical coding**, because it is simple, widely useful in practice, and it directly supports a transparent family of “zoom” transforms.

Extensions we do use

Hierarchical coding and “zooming” (FIL-ZOOM)

Motivation

In any qualitative coding process there is a tension between **detail** (keeping what respondents actually said) and **summarisation** (making patterns visible). Hierarchical factor labels give us a simple way to keep both:

- We can keep a detailed factor such as **Healthy behaviour; hand washing**.
- We can also treat it as an instance of a higher-level causal factor **Healthy behaviour** when we want a higher-level view.

This is particularly useful in causal mapping because it lets us simplify a complex map *without changing the underlying link evidence*: we are simply rewriting labels into more general ones.

Coding conventions

We use the separator ; to create nested factor labels, using a template like:

General concept; specific concept

For example:

New intervention; midwife training -> Healthy behaviour; hand washing

This is an example of what we called earlier "letting the labels do all the work". As we do not in any case have a native factors table where we might record actual relationships between say lower-level and higher-level factors, we can do the same thing implicitly simply with the text of the label.

In AI-assisted coding, a practical way to encourage hierarchical labels is to explicitly instruct the coder (human or AI) to label each factor using the template “**general concept; specific concept**”, and to always provide a verbatim quote for each coded claim so every link remains checkable.

For example, an AI might produce a single coded link such as:

- **Improved agricultural practices; diversified crops -> Increased income generation; more sales**
- Quote: “with a lot of good product, we are now able to sell more.”

You can read the separator as:

- “A, and in particular B” (forward), or
- “B, which is an example of / part of A” (reverse).

This can be extended to multiple levels:

Improved hospital skills training; new midwife training; hand washing instructions improved

Two practical conveniences follow immediately:

- Searching for the higher-level label (e.g. “Healthy behaviour”) will also find its nested sublabels.
- Higher-level labels can be created and changed on the fly (they are just the visible prefixes before ;), without maintaining a separate “parent code” structure.

Zooming and visualisation

The basic idea of a zoom view is: **rewrite nested labels to a chosen level of abstraction**, then draw the map using the rewritten labels.

At “zoom level 1” we simply truncate at the first ;:

- Healthy behaviour; hand washing becomes Healthy behaviour.

So if we have (at the detailed level):

New intervention; midwife training -> Healthy behaviour; hand washing

then we can show higher-level summary links such as:

- New intervention -> Healthy behaviour; hand washing
- New intervention; midwife training -> Healthy behaviour
- and (at the coarsest level) New intervention -> Healthy behaviour

Intuitively: by writing A; B you are explicitly licensing the interpretation “for high-level summary purposes, treat this as A”.

A worked example (one extension, end-to-end)

Suppose at the detailed level we code:

- New intervention; midwife training -> Healthy behaviour; hand washing

Then a zoom transform that truncates labels at the first ; (zoom level 1) yields a higher-level summary view:

- New intervention -> Healthy behaviour

Crucially, zooming is not “inference about the world”; it is a **label rewrite** that produces a simpler derived view of the same evidence.

You can also see this as a family of conservative “deductions” licensed by the ; separator. From:

- New intervention; midwife training -> Healthy behaviour; hand washing

we are explicitly allowing ourselves (for summary purposes) to treat it also as evidence for:

- New intervention -> Healthy behaviour; hand washing
- New intervention; midwife training -> Healthy behaviour
- New intervention -> Healthy behaviour

Caveats (don’t use the ; separator mechanically)

The ; separator encodes a commitment: that the more specific factor can be “rolled up” into the more general one without essentially distorting the causal story. So:

- A factor cannot belong to **two different** hierarchies at once (because there is no single parent to roll up into).
- Higher-level factors should usually be **semi-quantitative causal factors** in their own right (not mere themes). If “Health” is just a topic tag, use a tag/hashtag convention rather than `Health; X`.
- Try not to mix desirability within one hierarchy (e.g. avoid `Stakeholder capacity; lack of skills`, because zooming out would treat it as evidence for “Stakeholder capacity”). Use opposites coding (`~...`) for this kind of case.

When *not* to use a hierarchy (use a tag instead)

Do not use ; merely to group factors into a non-causal topic theme. For example, these are both “health-related” but are not naturally special cases of a single semi-quantitative causal factor:

- `Vaccinations law is passed`
- `Mortality rate`

In that kind of case, prefer a light tagging convention (e.g. `#health` or `(health)`) rather than forcing a hierarchy like `Health; vaccinations law is passed`.

Other analysis filters (stubs; treated elsewhere)

These are best understood as steps in an explicit pipeline (as in the formalisation paper) and we will not develop them here.

Context filters (FIL-CTX)

Restrict the evidence base by selecting sources (e.g. only women; only a time period; only a subgroup), then carry out the same downstream analysis on the restricted links table.

Frequency / evidence-threshold filters (FIL-FREQUENCY)

Retain only links meeting a threshold of evidence strength (e.g. `min_source_count=2`). This is about *evidence volume/breadth in the corpus*, not effect size.

Topological / path-tracing filters (FIL-TOPO)

Retain only links that sit on paths connecting chosen factors (e.g. “mechanisms linking Training to Yield”), typically with thread tracing constraints.

Bundling (FIL-BUNDLE)

Compute evidence-strength columns for each `Cause -> Effect` pair (e.g. `Citation_Count`, `Source_Count`) and use those in map/table views.

Opposites coding (FIL-OPP) (stub)

Rather than encoding signed/weighted edges, we can sometimes get most of the benefit by treating explicit opposites in labels (e.g. `~Employment` vs `Employment`) as a label-level device with simple inference and visualisation rules.

AI extensions (optional; stubs)

Soft recoding (magnets) (FIL-SOFT)

Many raw in-vivo labels can be aggregated by mapping them to a smaller vocabulary of “magnets” using semantic similarity. This changes labels (a recoding step), not the underlying minimalist meaning of a single coded link.

Auto recoding (clustering) (FIL-AUTO)

Alternatively, labels can be clustered into emergent groups and then rewritten accordingly, again as a recoding/aggregation step rather than a new causal logic.

How other schools of causal mapping extend minimalist links

The “minimalist” stance in this paper is deliberately radical: we code *undifferentiated* causal influence claims with provenance, and then do most interpretation work via filters and views. Many other causal mapping traditions extend the representation in the opposite direction: they add more **semantic structure to factors and links**, especially **polarity** (and sometimes strength/weights), often with an implicit assumption that factors behave like **variables** and that a link expresses some kind of monotonic relationship. For a broader overview of these traditions and their differences, see (Powell et al., 2024).

What these extensions usually add

- **Polarity (+/-)**: a link can mean “more of X leads to more of Y” (positive) or “more of X leads to less of Y” (negative).
- **Strength / weights**: a link can be assigned a magnitude (sometimes elicited, sometimes assigned by analysts), which invites quantitative reasoning and simulation-like interpretations.

- **Variable semantics and counterfactual clarity:** polarity and weights usually presuppose that the endpoints are comparable as variables (or at least as ordered states), so that “more/less” (or “increase/decrease”) is meaningful.

These extensions can be extremely useful in settings where respondents are explicitly reasoning in those terms, or where the purpose is modelling/simulation. But they also raise the bar: if we write down a signed or weighted link, we are committing not just to “X influenced Y”, but to a stronger claim about *how* X and Y vary and how changes propagate.

Axelrod-style cognitive mapping (signed causal beliefs)

In the Axelrod tradition, cognitive maps are often treated as representations of beliefs about causal influence among concepts, and they are frequently coded with some notion of **positive/negative influence** in addition to direction (Axelrod, 1976); (Axelrod, 1976). This makes it easier to talk about reinforcing/balancing structure and about the direction of change, but it also moves the representation closer to variables-with-values (even if the original text/elicitation was not precise about scales or functional form).

Eden & Ackermann cause mapping / problem structuring

The Eden/Ackermann “cause maps” tradition (including its software lineage) emphasises causal mapping as a practical tool for structuring messy problems and supporting decision making, often built interactively with respondents and iterated in workshops (Eden et al., 1992); (Ackermann et al., 2004); (Ackermann et al., 1996). Polarity and more explicit causal typing are more natural here because the map is typically negotiated in context (so the group can decide what is meant by “increase/decrease”, and can revise labels until the signed links make sense).

Comparative causal mapping (standardisation and cross-map comparison)

Comparative approaches focus on how to elicit, standardise, and compare large numbers of maps across people, groups, or time. This tends to bring in more explicit conventions for coding and comparison, and often assumes a more “variable-like” interpretation of factors so that maps can be aligned and analysed at scale (Laukkanen, 1994); (Laukkanen, 2012); (Markiczy & Goldberg, 1995); (Hodgkinson et al., 2004).

What it would take to extend our logic to signed links (and why we treat it separately)

We *can* extend our links-table logic to incorporate polarity, but doing so is not just “adding a column”; it changes the semantics.

At minimum, we would need:

- A **Polarity** field per coded link (e.g. +, -, **unknown**) that is grounded in the source text or elicitation.
- A rule for how polarity is inferred when it is implicit, ambiguous, or mixed (which is common in narrative material).

- A clear semantics for what **+** and **-** *mean* (typically: a monotonic relationship between variables), including what counts as “more/less” for a factor label.
- Aggregation rules for how to deal with disagreement and contradiction across sources, and for how to visualise those disagreements without creating spurious precision.

Because our target corpora typically do *not* make those commitments explicit (and because we are focused on modelling evidence-with-provenance rather than system dynamics), we do not treat signed links as part of the core method here. Where polarity matters in practice, we prefer to handle it with **label-level devices** (e.g. **~Employment** vs **Employment**) and with explicit, auditable transformations (e.g. **combine_opposites**), and we treat “signed/weighted link semantics” as a separate family of approaches to be discussed elsewhere.

What we cannot do

None of this really answers all the questions raised above about problematic cases such as what to do with “I got really tired because it was really hot”, and “I got really tired because it was really cold” or any other case where we have different factor codes which have shared information. At first blush, this isn't a problem, we can simply code “it was really hot” and “it was really cold” separately, but how to parse the contents to reflect the fact that these two are related? Or, how to parse the contents of “Improved health behaviour (hand washing)” and “Improved health behaviour (using a mosquito net)” to reflect the fact that they are somehow neighbours? We do have some tricks for this, but that would take us beyond the present discussion.

Causation about causation: enablers and blockers as second-order causal claims

One natural way to try to go beyond truly minimalist coding is to say that some claims are not about simple factors causing other simple factors, but about one factor influencing (enabling or blocking) a *causal connection* between two other factors.

Consider an enabler-style statement:

I went on holiday to Spain expecting to enjoy it, and indeed I did particularly enjoy it because I remembered to take my phrase book.

In strictly minimalist coding, we might record two undifferentiated links:

- Going on holiday to Spain --> Enjoying the holiday
- Remembering to take my phrase book --> Enjoying the holiday

But arguably the second claim is not really “the phrase book caused enjoyment” in the same way as the first. Rather, it enabled the normal causal power of the holiday to produce enjoyment. A more explicit encoding would therefore be:

1. Going on holiday to Spain --> Enjoying the holiday

2. Remembering to take my phrase book --> (Going on holiday to Spain --> Enjoying the holiday)

Visually, (2) would be drawn as an arrow pointing to the middle of the arrow in (1). Semantically, it can be read as:

The phrase book enabled the causal link from X to Y, in virtue of its causal power to do so.

Blockers are closely related but show the asymmetry more starkly:

I went on holiday to Spain, but I forgot to take my phrase book so I didn't enjoy the holiday at all.

One can code:

- Forgetting to take my phrase book --> Not enjoying the holiday

However, our intuition that “going on holiday” should also be part of the story is hard to capture without moving to a second-order encoding. The blocker case is visually similar to the enabler case, but it seems to require adding the idea of a causal power to *stop* something: the blocker destroys or disables the causal power of X to bring about Y.

I do not claim we can do much with this at scale; but it is a clear way of stating what seems to be “missing” from purely first-order, undifferentiated links in these edge cases.

Causal packages / conjunctions

Another class of difficult cases is causal packages: claims in which some *combination* of factors is said to be needed for an effect.

For example, “you need an accelerant and a spark to set a fire” is not well represented by coding two separate links (“accelerant causes fire” and “spark causes fire”), because that misses the conjunctive structure. One can code the cause as a single phrase (e.g. “an accelerant and a spark”), but then the phrase is not parsable in a way that lets us relate it to other claims about accelerants or sparks on their own.

In principle one could introduce special syntax for conjunctions, but the moment we do so we are immediately pushed towards stronger, more model-like commitments (e.g. about truth tables, interaction effects, non-linear combination rules), and it is unclear how much such structures would recur with enough regularity in ordinary language corpora to justify that added complexity.

Blockers and enablers

Maybe we could ascend from formally weaker but numerically overwhelming minimalist-coded data to make other rich conclusions, in particular about enablers and blockers like the

headphones and the rain. However, I don't think this is really possible. In minimalist coding, at the level of individual claims, you can code "The headphones enabled James to answer the question in the Zoom call" as

The headphones --> James was able to answer the question in the Zoom call

... but we cannot easily get inside the *contents* of the effect. We might like to be able to code this as the effect of the headphones not on a simple causal factor but on *another causal connection*, namely between the question on the Zoom call and James' answer, but we do not have any way at the moment to do this. It might be possible to extend minimalist coding to cope with this, perhaps ending up with three factors (headphones, question, answer) and some new syntactic rules to code their relationship, and some corresponding new semantic rules to be able to deduce more things about these three factors, but I think this would be **missing the point**. I'm not sure what we could do with these kinds of subtle relationships at any scale. Let's guess that within a given corpus, five percent of causal claims are of this form: what are the chances of such claims then overlapping enough in content that we could then apply our new more specialised deduction rules in more than a handful of cases?

It might be the case that certain specific more sophisticated causal constructions become **part of ordinary language**. For example: "Her post mocking Farage went viral, so Farage was forced to respond". Here, the concept of *going viral* is perhaps a kind of shorthand for a quite sophisticated causal claim, yet it might be common enough for us to be able to usefully code it (and reason with it) using only unadulterated minimalist coding, without causally unpacking "her post went viral". So that's useful, and maybe it is even useful in building some kinds of mid-range theory, but without actually understanding or unpacking what "going viral" means.

So that's it, in a nutshell. Sorry to disappoint, James.

See also:

(Powell et al., 2024)

(Powell & Cabral, 2025)

(Britt et al., 2025)

(Powell et al., 2025)

(Remnant et al., 2025)

Selected references (APA 7; added for qualitative-methods positioning)

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77–101. <https://doi.org/10.1191/1478088706qp0630a>

Charmaz, K. (2014). *Constructing grounded theory* (2nd ed.). SAGE.

Gläser, J., & Laudel, G. (2013). Life with and without coding: Two methods for early-stage data analysis in qualitative research aiming at causal explanations. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 14(2). <https://www.qualitative-research.net/index.php/fqs/article/view/1886>

Gläser, J., & Laudel, G. (2019). The discovery of causal mechanisms: Extractive qualitative content analysis as a tool for process tracing. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 20(3). <https://doi.org/10.17169/fqs-20.3.3386>

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. SAGE.

Mayring, P. (2000). Qualitative content analysis. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 1(2), Art. 20. <https://www.qualitative-research.net/index.php/fqs/article/view/1089>

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis: A methods sourcebook* (3rd ed.). SAGE.

Saldaña, J. (2021). *The coding manual for qualitative researchers* (4th ed.). SAGE.

[Note: we will add at least the formal codes like combine_opposites from the formalism paper]

References

Ackermann, Jones, Sweeney, & Eden (1996). *Decision Explorer: User Guide*. <https://banxia.com/pdf/de/DEGuide.pdf>.

Ackermann, Eden, & Cropper (2004). *Getting Started with Cognitive Mapping*.

Axelrod (1976). *Structure of Decision: The Cognitive Maps of Political Elites*. Princeton university press.

Axelrod (1976). *The Analysis of Cognitive Maps*. In *Structure of Decision : The Cognitive Maps of Political Elites*.

Barbrook-Johnson, & Penn (2022). *Participatory Systems Mapping*. In *Systems Mapping: How to Build and Use Causal Models of Systems*. https://doi.org/10.1007/978-3-031-01919-7_5.

Britt, Powell, & Cabral (2025). *Strengthening Outcome Harvesting with AI-assisted Causal Mapping*. https://5a867cea-2d96-4383-acf1-7bc3d406cdeb.usrfiles.com/ugd/5a867c_ad000813c80747baa85c7bd5ffafo442.pdf.

Copestake, Morsink, & Remnant (2019). *Attributing Development Impact: The Qualitative Impact Protocol Case Book*. March 21, Online.

Eden, Ackermann, & Cropper (1992). *The Analysis of Cause Maps*. <https://onlinelibrary.wiley.com/doi/10.1111/j.1467-6486.1992.tb00667.x>.

Hodgkinson, Maule, & Bown (2004). *Causal Cognitive Mapping in the Organizational Strategy Field: A Comparison of Alternative Elicitation Procedures*.

Laukkanen (1994). *Comparative Cause Mapping of Organizational Cognitions*.

Laukkanen (2012). *Comparative Causal Mapping and CMAP3 Software in Qualitative Studies*.
<https://doi.org/10.17169/fqs-13.2.1846>.

Markiczy, & Goldberg (1995). *A Method for Eliciting and Comparing Causal Maps*. Sage Publications Sage CA: Thousand Oaks, CA.

Powell, Larquemin, Copestake, Remnant, & Avarð (2023). *Does Our Theory Match Your Theory? Theories of Change and Causal Maps in Ghana*. In *Strategic Thinking, Design and the Theory of Change. A Framework for Designing Impactful and Transformational Social Interventions*.

Powell, Copestake, & Remnant (2024). *Causal Mapping for Evaluators*.
<https://doi.org/10.1177/13563890231196601>.

Powell, Cabral, & Mishan (2025). *A Workflow for Collecting and Understanding Stories at Scale, Supported by Artificial Intelligence*. SAGE Publications Sage UK: London, England.
<https://doi.org/10.1177/13563890251328640>.

Powell, & Cabral (2025). *AI-assisted Causal Mapping: A Validation Study*. Routledge.
<https://www.tandfonline.com/doi/abs/10.1080/13645579.2025.2591157>.

Remnant, Copestake, Powell, & Channon (2025). *Qualitative Causal Mapping in Evaluations*. In *Handbook of Health Services Evaluation: Theories, Methods and Innovative Practices*.
https://doi.org/10.1007/978-3-031-87869-5_12.

A simple measure of the goodness of fit of a causal theory to a text corpus

Abstract

Draft for an IJSRM submission. This paper proposes a lightweight grammar and logic for encoding, aggregating, and querying causal claims found in qualitative text data.

The specification is grounded in a "Minimalist" (or "Barefoot") approach to causal coding: it prioritises capturing the explicit causal claims made by sources, without imposing complex theoretical frameworks that may not align with how people naturally speak.

Intended audience: methodologically minded evaluators / qualitative researchers, and AI/NLP readers who want a precise target representation for “causal claims in text” (and a clear separation between evidence storage vs downstream interpretation).

Unique contribution (what this paper adds):

- A compact **data model** for causal evidence in text (**LinksTable**, optional **SourcesTable**, derived **Factors**) with an explicit evidence constraint (quotes as substrings of source text when sources are present).
- A set of **syntax + semantics rules** for what counts as a valid coded claim and what can (and cannot) be inferred when aggregating claims across sources.
- A “pipeline” view of analysis as **operations on a links table**, which is designed to stay compatible with minimalist coding and to avoid accidental drift into system-model semantics (e.g., polarity/weights-by-default).

What does one piece of causal coding *mean*?

- We have syntactic rules to say what is a valid piece of causal coding.
- We have syntactic rules to say how to combine them into causal maps.
- We have syntactic rules for operations on causal maps.

For each of these rules we need to also give semantic rules to say what it also means, or how it combines meanings of its component parts.

The basic coding rule is something like:

Look for places where it is stated or claimed that one thing causes or influences another.

Causal mapping logic as rules about relationship between texts is particularly relevant in the age of LLMs.

Having a causal influence is a lot weaker and more easily than causally affecting. So a mitigating action can causally influence something but not make it happen.

Project context (companion papers)

- Narrative coding stance and limits: [Minimalist coding for causal mapping](#)
- QDA positioning: [Causal mapping as causal QDA](#)
- Practical transforms over a links table: [Magnetisation](#); [Combining opposites, sentiment and despite-claims](#); [Hierarchical coding](#)
- Coverage-style fit diagnostic: [A simple measure of the goodness of fit of a causal theory to a text corpus](#)

1. Data Structures

The foundation of the specification is the **Project Data** package, which strictly separates the causal claims from the source material.

Definition Rule DS-PROJECTDATA: Project Data

- **Definition:** `ProjectData = LinksTable + optional SourcesTable`
- **Note:** `SourcesTable` may be omitted (or empty). If it is present, the evidence constraint in DS-LINKS-SOURCES applies.

Syntax Rule DS-LINKS: The Basic Links Table

A list of causal connections where each row represents one atomic claim.

- **Structure:** A table containing at least the following columns:
- **Cause:** Text label for the driver (influence factor).
- **Effect:** Text label for the outcome (consequence factor).
- **Link_ID:** Unique identifier for bookkeeping.
- **Context:** Optional identifier for distinguishing contexts.
- **Source_ID:** Identifier for the origin of the claim (e.g., document ID, participant ID).
- **Extensions:** The table may contain additional columns (e.g., `Sentiment`, `Time_Period`) or tags.

Syntax Rule DS-SOURCES: The Sources Table (optional)

A registry of documents, interviews, or respondents.

- **Structure:** A table containing:
- **Source_ID:** Unique key.
- **Full_Text:** The complete text of the source document.
- **Metadata:** Optional custom columns representing attributes of the source (e.g., Gender, Region, Date).

Syntax Rule DS-LINKS-SOURCES: Evidence Constraint (if Sources table is present)

- **Additional column:** The Links table also contains:
- **Quote:** The specific text segment evidencing the claim.

- **Constraint:** If a Sources table is provided, then for every link **L** in the Links table:
- **L.Source_ID** MUST match a **Sources.Source_ID**, and
- **L.Quote** MUST be a substring of **Sources.Full_Text** for the matching **Source_ID**.

Coding is strictly evidence-based.

Definition Rule DS-FACTORS: The Factors Derivation

There is no independent table for Factors stored in the Project. Factors are derived entities.

- **Definition:** **Factors** = unique values in **LinksTable.Cause** + **LinksTable.Effect**
- **Note:** A "Factors Table" is a transient structure created only during analysis.

2. Coding and Interpretation

This section defines how text is translated into the data structures defined above.

Definition Rule COD-DEF: The Definition of Coding

Coding is the process of extracting links from source text into the Links table.

Interpretation Rule COD-ATOM: The Atomic Causal Claim

A single row in the Links table, **Link | Cause="A" | Effect="B" | Source_ID="S1"**, is interpreted as:

- *"Source S1 claims that A causally influenced B in virtue of its causal power to do so."*

Interpretation Rule COD-BARE: Bare Causation

The relationship **A -> B** implies **influence**, not determination.

- **Negative Definition:** It does NOT imply **A** is the only cause of **B**.
- **Negative Definition:** It does NOT imply **A** is sufficient for **B**.

Example:

- **Source Text:** "Because of the drought, we had to sell our livestock." (from Source 12)
- **Coded Link:**
- Cause: **Drought**
- Effect: **Selling livestock**
- Source_ID: **12**
- Quote: "Because of the drought, we had to sell our livestock"

Interpretation Rule COD-MIN: Minimalist Coding Principles

The coding schema prioritizes explicit claims over complex theoretical frameworks.

- **The 90% Rule:** Code the simple form "**X causally influenced Y**". Explicit coding of complex logic (enablers, sufficiency, non-linearity) is not provided for.

- **Propositions, Not Variables:** Factors are simple propositions (e.g., "Jo started shouting"), not variables with values. Distinct concepts should not be merged prematurely.
- **Example (why this matters):** Code **Poverty** and **Wealth** as distinct factors rather than values of one "Economic Status" variable. A source may claim **Poverty** -> **Stress** and also **Wealth** -> **Investment**; collapsing these too early can obscure distinct narratives.
- **No Absences:** Do not code absences. If a source does not mention a factor, it is unknown, not absent.
- **Realism & Partiality:** "X influenced Y" means X had the causal power to affect Y in this context. It implies a contribution, not total determination.

Surface cognition

- **Goal:** Model the speaker's expressed causal thinking (cognition), not necessarily underlying objective reality.
- **Method:** Code only what is explicitly said; avoid inferring hidden variables or unstated counterfactuals.

Interpretation Rule COD-MULTI: The meaning of multiple links

A table containing multiple links simply asserts the logical conjunction of the links:

- Source S1 claims that A causally influenced B in virtue of its causal power to do so.
- Source S1 claims that C causally influenced D in virtue of its causal power to do so.
- Source S2 claims that A causally influenced C in virtue of its causal power to do so.

So, unless specific contexts are specified, if Source S1 claims that **A** -> **C** and also that **B** -> **C**, this is neither a contradiction nor (by default) a claim that A-and-B jointly influenced C as a package. It is simply two separate claims.

So if families are talking about reasons for family disputes, and family F mentions social media use, and family G mentions homework, we do not usually interpret this as meaning that family F does *not* think that homework can also be a cause of family disputes.

3. The Filter Pipeline (Query Language)

Analysis is performed by passing a links table through a sequence of filters.

Syntax Rule FIL-PIPE: The Interpretation Pipeline

The interpretation of a result is defined by the cumulative restrictions or transformations of the filters applied.

- **Syntax:** **Input** |> **Filter1** |> **Filter2** |> **Output**
- **Multi-line Syntax:**

Input

```
|> Filter1  
|> Filter2  
|> Output
```

- **Processing:** Filters are applied sequentially. The output of **Filter N** becomes the input of **Filter N+1**.

Types of Filters

- **All filters take a Links table as input.** Most filters return a Links table (so they can be chained). **Output filters terminate the pipeline** by returning a derived view (table/summary) rather than a Links table.

In practice (as in the app), filter behaviour is often **multi-effect**. For example, a filter may rewrite labels *and* add tracking columns. So instead of forcing each filter into exactly one “type”, we treat each filter as having an **effect signature**:

- **Row selection:** changes *which links* are included (drops/retains rows).
- **Label rewrite:** changes **Cause/Effect** labels (recoding/normalisation).
- **Column enrichment:** adds or recalculates columns (metadata/metrics/flags), typically to support later filtering or display.
- **Configuration:** changes display/formatting settings without changing the links table (app-specific; not formalised here as a core links-table transform).

Below, filters are grouped by their **primary intent**, and each rule declares its **Effects:** line.

Row-selection filters

Syntax Rule FIL-CTX: Context Filters

Reduces the evidentiary base based on Source metadata.

- **Effects:** row selection
- **Operation:** `filter_sources | <criteria...>`
- **Interpretation:** Restrict the evidence base to links whose **Source_ID** is in the retained set of sources.

Syntax Rule FIL-FREQUENCY: Content Filters

Reduces the evidentiary base based on signal strength.

- **Effects:** row selection
- **Operation:** `filter_links | <criteria...>`
- **Interpretation:** Retain only links meeting an evidence threshold (e.g., `min_source_count=2`).

Syntax Rule FIL-TOPO: Topological Filters

Retains links based on their position in a causal chain.

- **Effects:** row selection
- **Operation:** `trace_paths | from="<factor>" | to="<factor>" | <options...>`
- **Interpretation:** "Retaining only mechanisms that connect *From* to *To*."

Label-rewrite filters

Interpretation Rule FIL-ZOOM: The Zoom Filter (Hierarchical Syntax)

Extends the logic to handle nested concepts via a separator syntax.

- **Effects:** label rewrite
- **Syntax:** Factors may use the ; separator (e.g., `General Concept; Specific Concept`).
- **Interpretation:** `A; B` implies `B` is an instance or sub-component of `A`.
- **Operation:** `transform_labels | zoom_level=1`. If `zoom_level=1`, rewrite labels by truncating text after the first separator.
- **Inference:** At Zoom Level 1, `A; B` is treated logically as `A`.

Interpretation Rule FIL-OPP: The Combine Opposites Filter (Bivalence Syntax)

Extends the logic to handle polarity/negation.

- **Effects:** label rewrite; column enrichment
- **Syntax:** Factors may use the ~ prefix (e.g., `~Employment`) or tag pairs.
- **Interpretation:** `~A` is the negation of `A`.
- **Operation:** `combine_opposites`. Rewrites negative labels (e.g., `~Income`) to their positive counterparts (`Income`) and adds tracking columns such as `flipped_cause` and `flipped_effect`.
- **Inference:** Evidence for `~A -> ~B` is treated as corroborating evidence for `A -> B` (with flipped polarity).

Column-enrichment filters

Syntax Rule FIL-BUNDLE: The Bundling Filter

This filter aggregates co-terminal links (links with the same cause and effect) to calculate evidence metrics without reducing the row count. We normally think of it as being automatically applied after any other filter.

- **Effects:** column enrichment
- **Operation:** `bundle_links`
- **Definition (bundle object):** `Bundle(A, B) = all links L where L.Cause <mark> A AND L.Effect </mark> B`
- **Logic:** For every link `L`, identify the set of all links `S` where `S.Cause <mark> L.Cause AND S.Effect </mark> L.Effect`.
- **Transformation:** Appends new columns to the Links table:

- **Bundle:** For convenience, a text representation of the connection (e.g., "A -> B").
- **Citation_Count:** The total count of rows in set *S*. Represents volume of coding.
- **Source_Count:** The number of unique *Source_IDs* in set *S*. Represents breadth of evidence (consensus).
- **Lemma:** *Source_Count* <= *Citation_Count*.

We measure importance using two distinct metrics:

- **Citation Count:** The total number of times a link or factor was mentioned across the entire project. This counts every single row in the data.
- *Technical:* *citation_count*
- **Source Count (or Number of People):** The number of *unique* sources (people or documents) that mentioned a link or factor. This avoids double-counting if one person repeats the same point multiple times.
- *Technical:* *source_count*

Output filters

Output Rule OUT-FACTORS: Factors table view

Returns a Factors table (one row per factor) derived from a Links table (typically after *FIL-BUNDLE*).

- **Operation:** *factors_view*
- **Interpretation:** Aggregate over the set of factor labels appearing anywhere in *Cause* or *Effect*, and compute per-factor summaries (e.g., role metrics).

Output Rule OUT-MAP: Graphical map view

Returns a graphical network view of the current Links table.

- **Operation:** *map_view*
- **Interpretation (data):**
- **Nodes:** Factors (labels appearing in *Cause* or *Effect*).
- **Edges: Bundles** (one edge per *Cause* -> *Effect* pair), built from the **current filtered/transformed labels** (so the map reflects Zoom/Combine-Opposites/etc.).
- **Bundling:** If bundle-level columns are not already present, the map view implicitly applies *FIL-BUNDLE* to compute bundle metrics (e.g., *Citation_Count*, *Source_Count*) used for labels and styling.
- **Interpretation (presentation):** The view includes a formatting configuration that maps derived metrics to visual encodings (e.g., edge width by citation/source count; edge colour/arrowheads by mean sentiment; node colour by outcomeness; node/label sizes by frequency), plus a legend summarising the current view and applied filters.

Definition Rule MET-NODE: Factor Role Metrics

These metrics describe the topological role of a factor.

- **In-Degree (incoming citations):** Count of incoming links (times this factor appears as an Effect).
- *Technical:* `citation_count_in`
- **Out-Degree (outgoing citations):** Count of outgoing links (times this factor appears as a Cause).
- *Technical:* `citation_count_out`
- **Outcomeness:** $\text{In-Degree} / (\text{In-Degree} + \text{Out-Degree})$.
- *Interpretation:* A score nearing 1 indicates an Outcome; a score nearing 0 indicates a Driver.

4. Example Queries

Example A: The "Drivers" Query Question: *What do female participants say are the main drivers of Income?*

```
Result = ProjectData
  |> filter_sources | Gender="Female"           // Rule FIL-CTX
  |> trace_paths | to="Income" | steps=1        // Rule FIL-TOPO
  |> filter_links | min_citations=2             // Rule FIL-FREQUENCY
```

Example B: The "Mechanism" Query Question: *Is there valid narrative evidence that Training leads to Better Yields?*

```
Result = ProjectData
  |> transform_labels | zoom_level=1             // Rule FIL-ZOOM
  |> trace_paths | from="Training" | to="Yield" | thread_tracing=TRUE // Rule FIL-TOPO
```

5 Causal Inference?

Inference Rule INF-EVID: Evidence is not effect size

We quantify **evidence strength**, not **causal effect strength**.

- **Observation:** `Link | Cause="A" | Effect="B"` appears 10 times.
- **Inference:** There are 10 pieces of evidence (10 coded mentions) for the claim `A -> B`.
- **Invalid inference:** The influence of A on B is 10 times stronger than a link appearing once.

Inference Rule INF-FACT: Factual Implication

If we observe `Link | Cause="A" | Effect="B" | Source_ID="S1"`:

- **Deduction:** `Source S1` claims `A` happened/exists.
- **Deduction:** `Source S1` claims `B` happened/exists.

Inference Rule INF-THREAD: Thread Tracing (Valid Transitivity)

We can infer a long causal chain (indirect influence) only if one source provides every step.

- **Logic:** Link | Cause="A" | Effect="B" | Source_ID="S1" AND Link | Cause="B" | Effect="C" | Source_ID="S1" => Valid path A -> B -> C.

Inference Rule INF-CTX: The Context Rule (The Transitivity Trap)

We cannot infer causal chains by stitching together different sources without checking context.

- **Logic:** Link | Cause="A" | Effect="B" | Source_ID="S1" AND Link | Cause="B" | Effect="C" | Source_ID="S2" => INVALID path A -> C, unless S1 and S2 share the same Context.

Context

These questions may depend on a somewhat hidden assumption: that all the causal claims (the links) come from a single context. Such as, in most cases, when all the claims are all agreed on by a group as in participatory systems mapping (PSM). For example, when we wrote "which factors are reported as being causally central?", can we really answer that by simply checking the network? From:

Factor X is central within these claims

Can we deduce:

Factor X is claimed to be central

Or, can we go from:

There are two relatively separate groups of causal claims?

Can we deduce:

It is claimed that there are two relatively separate groups of causal claims?

In general, no. It is easy to think of counter-examples. If we ask a parents and children about the causal network surrounding family disputes, we might get two relatively separate causal networks with only a little overlap. From this we cannot conclude that these respondents taken together claim that there are two relatively separate systems. It might be that the parents and children indeed are giving information about relatively separate systems about which they each have the

best information, or it might be that the two groups are telling conflicting and perhaps incompatible stories.

We could express this as, say, the first axiom of causal mapping:

If a network of causal evidence from context C has property P, we can conclude that there is evidence that the corresponding causal network has property P, but again only in context C.

$P(E(N)) \rightarrow E(P(N))$

We often assume that contexts are sources and sources are contexts. But this is not always the case. For example one respondent might give two sets of information, one from before losing their job and one set from afterwards, without trying to encode the job loss as a causal factor within the network of claims.

In a PSM workshop, there may be multiple respondents but, as long as they construct a consensus map, these are all treated as one source. Part of the job of the moderator is also to ensure that the claims (evidence) all come from one context, which is the same as saying: we can validly make inferences like those above. I don't know whether PSM moderators actually do this.

You might say "this is all pointless because it depends what you mean by context", and that is exactly true. All we have done is

Appendix A: AI Extensions

These filters extend the core logic using probabilistic AI models (Embeddings or Clustering).

Interpretation Rule FIL-SOFT: The Soft Recode Filter

Extends interpretation logic using semantic similarity (vector embeddings).

- **Operation:** `soft_recode` | `magnets="<...>"` | `similarity_threshold="<...>"`
- **Inference:** If `Label A` is similar to `Magnet M` (> threshold), treat `A` as `M`.

Interpretation Rule FIL-AUTO: The Auto Recode Filter

Extends logic using unsupervised clustering.

- **Operation:** `auto_recode` | `target_clusters=K`
- **Inference:** Factors group together based on inherent semantic proximity into `K` emergent themes.

Plain coding

A simple measure of the goodness of fit of a causal theory to a text corpus

Abstract

This guide addresses a cluster of **tricky but practical problems** in causal coding: how to represent **oppositeness**, **sentiment/valence**, and **“despite” constructions** in a way that stays close to ordinary language and remains auditable.

Many causal mapping and systems traditions address these issues by quickly “variablising”: treating factor labels as variables, and links as signed (and sometimes weighted) relationships. That can be powerful, but it also introduces strong extra commitments (about polarity, scales, functional form, and what counts as “more/less”) that are often not warranted by ordinary narrative text.

Instead we take a **piece-by-piece approach**:

1. We start with **combining opposites** as a conservative label convention plus an explicit transform over a links table.
2. We then “turn 45 degrees” to the different-but-overlapping problem of **sentiment**, which becomes especially useful/necessary when doing AI-assisted coding and embedding-based aggregation. Sentiment and opposites are hard to combine cleanly.
3. We then introduce **“despite” coding** for countervailing conditions (“E happened despite Z”) without pretending that Z “caused” E in the ordinary way.

We end by noting some hard cases where these systems collide.

See also: [Minimalist coding for causal mapping](#); [A formalisation of causal mapping](#); [Magnetisation](#); [A simple measure of the goodness of fit of a causal theory to a text corpus](#).

Intended audience: practitioners doing causal coding from narrative text (especially at scale / with AI assistance) who keep running into polarity/valence edge cases.

Unique contribution (what this guide adds):

- A conservative **opposites convention + transform** (combine opposites while retaining flip status).
- A clear separation between **oppositeness** and **sentiment** (why they overlap in practice but don’t collapse cleanly).
- A scalable encoding for **“despite”** clauses as a link type/tag, rather than mis-coding them as ordinary causes.

Introduction

In the first part of this guide we dealt only with undifferentiated causal links which simply say “C influenced E”, or more precisely: “Source S claims/believes that C influenced E.” This is the

barebones representation: a links table whose rows are individual causal claims with provenance (source id + quote) and whose columns include at minimum **Cause** and **Effect** labels.

Barebones causal links are commonly used in (at least) two ways:

- **Event-claim reading (QuIP-style)**: interpret a link as a claim about a past episode (“C happened; E happened; C made a difference to E”), with an open question of how far it generalises.
- **Factor-relation reading**: treat links as claims about influence relations among factors, without committing to what happened in any specific case.

In Part 1 we also introduced **hierarchical factor labels** using the **;** separator, where **C; D** can be read as “D, an example of / instance of C”, and can later be **rewritten** (zoomed) to a higher level by truncating the label.

This guide (Part 2) adds three extensions that remain compatible with barebones link coding:

- **Opposites conventions** in factor labels (a label-level device).
- **Sentiment/valence** as an additional annotation layer (useful especially with AI coding).
- **Despite coding** to capture countervailing conditions without misrepresenting them as ordinary causes.

We will describe the conventions in app-independent terms. (The Causal Map app happens to implement these ideas as part of a standard “filter pipeline”, but the logic is not app-specific.)

Combining opposites

The problem

In everyday coding, we often end up with “opposite” factors like:

- **Employment** vs **Unemployment**
- **Good health** vs **Poor health**
- **Fit** vs **Not fit**

If we keep these as unrelated labels, we make downstream analysis harder. For example:

- When we query for “health”, we may miss evidence coded as “illness”.
- We cannot easily compare (or combine) evidence for **Fit -> Happy** with evidence for **Not fit -> Not happy** without manually re-aligning them.

Many causal mapping traditions solve this by treating factors as variables with signed links. Here we describe a simpler alternative that stays close to ordinary language and avoids variable semantics “by default”.

The convention: mark opposites in labels

To signal that two factor labels are intended as opposites, use the **~** prefix:

- Y and $\sim Y$

We talk about **opposites** rather than plus/minus because this avoids implying valence or sentiment. For example:

- **Smoking** is the opposite of **\sim Smoking** (not smoking), but which one is “good” depends on context.

Non-hierarchical opposites are straightforward:

- **Eating vegetables**
- **\sim Eating vegetables**
- **Smoking**
- **\sim Smoking**

The useful part: apply a transform/filter to a links table (and/or a map view)

The convention above is only a convention until we do something with it. The next step is to apply an explicit **transform** to a links table (and then to whatever views are derived from it).

The transform is:

1. Detect opposite pairs that are present (both Y and $\sim Y$ appear in the current dataset).
2. Rewrite any occurrence of $\sim Y$ to Y .
3. Record, for each link endpoint, whether it was flipped (e.g. **flipped_cause**, **flipped_effect**).

After this transform:

- We can aggregate evidence for Y and $\sim Y$ under a single canonical factor label Y **without losing the original meaning**, because we still know which endpoints were flipped.
- A link has two local “polarities”: whether the cause label was flipped, and whether the effect label was flipped.
- If **exactly one** end is flipped, the overall relationship direction is “reversed” in the intuitive sense (compared to the unflipped link).
- If **both** ends are flipped, the overall relationship direction is not reversed, but the evidence is still distinct (it came from the opposite-on-both-ends claim).

Crucially: **no information is lost** by the transform, because flip-status remains attached to the evidence. You can always reconstruct the original statement-level content from the transformed table.

This general “apply transforms to a links table; then render a map/table view of the transformed data” is the same idea as the filter pipeline described in the user-guide material: filters are operations over a links table, and maps are derived views of the filtered/transformed links.

Opposites coding within a hierarchy

When using hierarchical labels (with ;), the ~ sign may appear:

- at the very start of the whole label, and/or
- at the start of any component within the label.

The same transform idea applies: to “combine opposites” we flip components so that opposites align at each hierarchical level.

Opposites within components of a hierarchy

Sometimes we need ~ within components, e.g.:

- Healthy habits; eating vegetables
- ~Healthy habits; ~eating vegetables

and:

- Healthy habits; ~smoking (not smoking is a healthy habit)
- ~Healthy habits; smoking (smoking is an unhealthy habit)

After combining opposites, these pairs can be aligned under shared canonical labels while retaining flip status per component (so we do not collapse “healthy” into “unhealthy” or vice versa by accident).

Bivalent variables?

Opposites coding is **not** the same as assuming that every factor is a bivalent variable (present vs absent). We are not claiming exhaustiveness: it is not the case that everything must be either **Wealthy** or **Poor**, and it is often wrong to treat “absence” as having causal powers.

Opposites coding is a practical device used only where:

- both poles occur naturally in the text and therefore in the coding, and
- it would usually be incoherent to apply both poles simultaneously in the same sense.

Which pairs of factors should we consider for opposites coding?

Use opposites coding for a pair of factors X and Y (i.e. recode Y as ~X or recode X as ~Y) when both occur in the data and are broadly opposites.

If in doubt about which member to treat as the canonical label, we usually pick X as the “primary” member if it is:

- usually considered as positive / beneficial / valuable, and/or
- usually associated with “more” of something rather than “less” of something.

Alternative convention (explicit opposite pairing)

Sometimes you may have pairs that are conceptually opposite but do **not** share a clean string form like Y vs $\sim Y$ (e.g. **Wealthy** vs **Poor**).

In that case, use an explicit pairing tag so that a deterministic transform can combine them later. For example:

- **Wealthy** [1]
- **Poor** [~ 1]

This makes the pairing unambiguous: **Poor** [~ 1] is declared to be the opposite of **Wealthy** [1]. A “combine opposites” transform can then rewrite the opposite-labelled item to the canonical label while recording flip status (exactly as described above). This is the same general idea as the “transform filters temporarily relabel factors” pattern in the filter pipeline documentation (see [content/999 Causal Map App/080 Analysis Filters Filter links tab \(\(filter-link-tab\)\).md](#)), but the logic is independent of any particular software.

What can we *do* with opposites once we have them?

Once opposites are marked (and optionally combined via an explicit transform), we can apply ordinary operations to a links table and then render useful views:

- **Querying:** searching for Y can intentionally retrieve both Y and $\sim Y$ evidence (depending on whether you search pre- or post-transform).
- **Aggregation without collapse:** you can summarise evidence under a canonical label Y while still distinguishing which claims involved the opposite sense via flip flags.
- **Visualisation:** you can render a map from the transformed links table and style links differently depending on whether the cause and/or effect endpoint was flipped, so viewers can see “this includes opposite-evidence” rather than mistaking it for ordinary evidence.

This “links table \rightarrow transforms \rightarrow map/table view” pattern is the same general idea as a filter pipeline (implemented in many tools; the Causal Map app is one).

Adding sentiment

Polarity of factor labels

There are challenges with coding and validating concepts which on the one hand could be seen to have polarity from a quantitative point of view and on the other hand may have positive or negative sentiment associated with them. Quantitative polarity and subjective sentiment often overlap in confusing ways. When coding, distinguishing between opposites like employment and unemployment is usually important: they can be viewed as opposites, but each pole has a distinct meaning which is more than just the absence of the other. We can call these “bipolar” concepts after [\(Goertz, 2020\)](#).

However, though employment and unemployment can be seen as opposites from a "close level" view, at a more general or abstract level, they could both fall under a category like economic issues. In the space of NLP embeddings, these two concepts may be surprisingly close.

For other pairs like not having enough to eat, and having too much to eat, there are multiple opposites (which may appear frequently as a causal factor) with an intervening zero (which may not be mentioned very often).

Coding these different kinds of concept pairs can be difficult and depends on use and context. For these and other reasons, our naive approach codes employment and unemployment as well as not having enough to eat, and having too much to eat separately.**

How to add sentiment?

You can now auto-code the sentiment of the consequence factor in each link.

You only have to do this once, and it takes a little while, so wait until you've finished coding all your links.

When you are ready, click on the File tab, and under the 'About this file', there's the 'ADD SENTIMENT' button. You just have to click on it and wait for the magic to happen




Design sem nome (1).png



So each claim (actually, the consequence of each claim) now has a sentiment, either -1, 0 or 1.

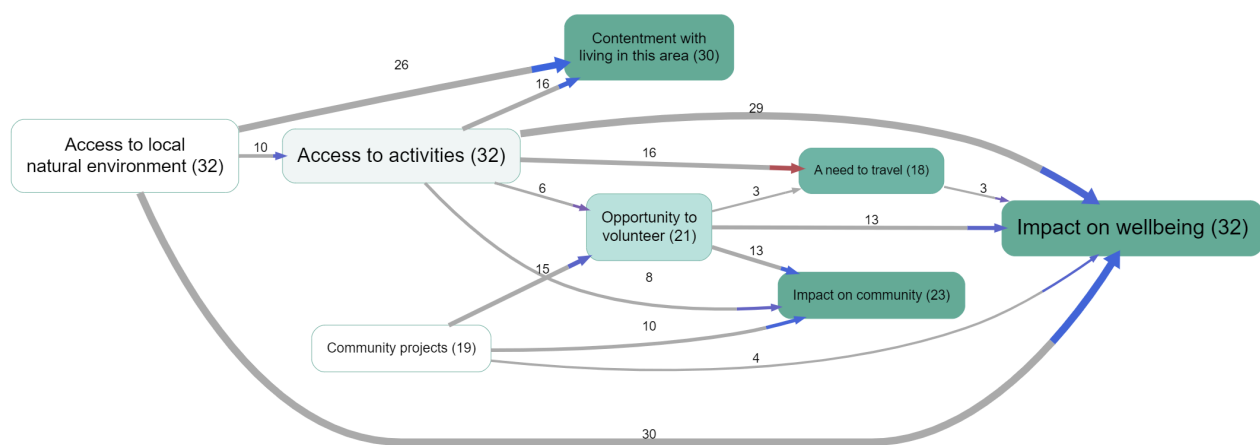
Many links are actually a bundle of different claims. We can calculate the sentiment of any bundle, as simply the average sentiment. So an average sentiment of -1 means that all the claims had negative sentiment. An average of zero means there were many neutral sentiments and/or the positive and negative sentiments cancel each other out.

Only the last part is coloured, because the colour only expresses the sentiment of the effect, not the cause.

Once you have autocoded sentiment for your file, you can switch it on or off using  [Formatters: Colour links](#).

Tip

When displaying sentiment like this, reduce possible confusing by making sure that you either use only neutral factor labels like Health rather than Good health or Improved Health: an exception is if you have *pairs* of non-neutral labels like both Poor health alongside Good health. You can do this either in your raw coding or using  [Transforms Filters: !\[\]\(cd41623988aee99b5b915fa19a633d9e_img.jpg\) Magnetic labels](#) or  [Canonical workflow](#)



Filename: solvacare. Citation coverage 25%; 610 of 2426 total citations and 32 of 32 total coded sources are shown here. Numbers on factors show source count.. Factor sizes show citation count. Darker factor colours show greater outcomeness. Numbers on links show source count.

Zooming in to level 1 of the hierarchy. Auto clustering factors using label set new. Top 8 factors by citation count. Showing only links with at least 3 sources.

Adding some colour: a discussion of the problem of visualising contrary meanings

The problem

We’ve already described our approach to making sense of texts at scale by almost-automatically coding the causal claims within them, encoding each claim (like “climate change means our crops are failing”) as a pair of factor labels (“climate change” and “our crops are failing”): this information is visualised as one link in a causal map. We use our “coding AI” to code most of the causal claims within a set of documents in this way. We have had good success doing this quickly and at scale without providing any kind of codebook: the AI is free to create whatever factor labels it wants.

There is one key remaining problem with this approach. Here is the background to the problem: if the coding AI is provided with very long texts, it tends to skip many of the causal claims in fact contained in the documents. Much shorter chunks of text work best. As we work without a codebook, this means that the AI produces hundreds of different factor labels which may overlap substantially in meaning. In turn this means that we have to cluster the labels in sets of similar meaning (using phrase embeddings and our “Clustering AI”) and find labels for the sets. This all works nicely.

But the problem is that, when we use phrase embeddings to make clusters of similar labels, seeming opposites are often placed quite close together. In the space of all possible meanings, unemployment and employment are quite close together – they would for example often appear on the same pages of a book – and both are quite far from a phrase like, say, “climate change”. But this is unsatisfactory because if in the raw text we had a link describing how someone lost their job, coded with an arrow leading to a factor unemployment alongside another piece of text describing how someone gained work, represented by an arrow pointing to employment if these

two labels are combined, say into employment or employment issues the items look very similar and we seem to have lost some essential piece of information.

Can't we use opposites coding?

In ordinary manual coding (see [+ - Opposites](#)) we solve this problem by marking some factors as contrary to others using our ~ notation (in which ~Employment can stand for Unemployment, Bad employment, etc) and this works well. However while it is possible to get the coding AI to code using this kind of notation, it is not part of ordinary language and is therefore not understood by the embeddings API: the ~ is simply ignored even more often than the difference between Employment and Unemployment. In order to stop factors like employment and unemployment ending up in the same cluster it is possible to exaggerate the difference between them by somehow rewriting employment as, say, “really really crass absence of employment” but this is also unsatisfactory (partly because all the factors like really really crass X tend to end up in the same cluster).

New solution

So our new solution is simply to accept the way the coding-AI uses ordinary language labels like employment and unemployment and to accept the way the embedding-AI clusters them together. Instead, we recapture the lost “negative” meaning with a third AI we call the “labelling AI”. This automatically codes the sentiment of each individual causal claim so that each link is given a sentiment of either +1, 0 or -1. For this third step we use a chat API. The instruction to this third AI is:

"I am going to show you a numbered list of causal claims, where different respondents said that one thing ('the cause') causally influenced another ('the effect') together with a verbatim quote from the respondent explaining how the cause led to the effect.

The claims are listed in this format: 'quote ((cause --> effect))'.

The claims and respondents are not related to one another so treat each separately.

For each claim, report its *sentiment*: does the respondent think that the effect produced by the cause is at least a bit good (+1), at least a little bad (-1) or neutral (0).

Consider only what the respondent thinks is good or bad, not what you think.

If you are not sure, use the neutral option (0).

NEVER skip any claims. Provide a sentiment for every claim."

The previous solution coloured the whole link which was fine in most cases but led to some really confusing and incorrect coding where the influence factor was involved in the opposite sense, as in Access to activities below. One might assume that the red links actually involve some kind of negative (or opposite?) access to activities, but we don't actually know that because it wasn't

coded as such. Other alternatives would be to also automatically separately code the sentiment of the first part of the arrow, but this doesn't work because sometimes the sentiment is not in fact negative. We would have to somehow automatically code whether the influence factor is meant in an opposite or contrary sense but this is hard to do.

"Despite" coding

The problem: countervailing conditions that “failed”

Narratives often contain claims of the form:

“E happened despite Z.”

Examples:

- “The heavy rains made the river levels rise, **despite** the prevention and clearance work.”
- “**Despite** me reminding him multiple times, he missed the train.”

In ordinary language, the “despite” clause does two things at once:

1. It signals that **Z was expected to prevent or reduce E** (a countervailing influence).
2. It asserts that **E happened anyway** (so Z was insufficient, absent, overridden, or ineffective in that case).

If we only code the main causal claim (e.g. **Heavy rains -> River levels rose**), we lose potentially important information about attempted mitigations, resistance, barriers, or protective factors.

But it is also usually wrong to code the “despite” clause as an ordinary positive causal link, e.g. **Prevention work -> River levels rose**, because that flips the intended meaning on its head.

The convention: a “despite” link type (or tag)

We therefore treat “despite” as a **link type**, not a different semantics for **->**.

A minimal encoding is:

- **Main link** (ordinary): **X -> E**
- **Despite link** (typed/tagged): **Z -despite-> E**

Equivalently (in a plain links table), keep the same **Cause** and **Effect** columns but add either:

- a **link_type** column with values like **normal** vs **despite**, or
- a **link_tags** column containing a tag like **#despite**.

The key idea is that **-despite->** does **not** mean “Z caused E”. It means:

“Z was presented as a countervailing influence against E, in virtue of its causal power to work against E, but E occurred anyway.”

What can we do with “despite” links?

Because the distinction is explicit in the links table, we can decide—per analysis—how to treat these links:

- **Visualise** them distinctly (different colour/line style), so the map shows “tensions” rather than silently dropping them or misreading them.
- **Filter:** show only **despite** links to see what people present as failed protections / resistances / mitigations.
- **Compare:** for similar outcomes, do some sources describe Z as effective (ordinary preventive claim, e.g. $Z \rightarrow \sim E$) while others describe “despite” failures ($Z \text{ -despite-} \rightarrow E$)? That contrast can be analytically important.
- **Count separately:** include them in evidence tallies only under a separate counter (e.g. **Despite_Citation_Count**) so you don’t accidentally inflate evidence for “Z causes E”.

Worked example

Text:

“The heavy rains made the river levels rise, despite the prevention and clearance work.”

Coding:

- Heavy rains \rightarrow River levels rose
- Prevention and clearance work -despite- River levels rose

This preserves the main mechanism while also storing the claim that a mitigation was present but insufficient.

Optional background: force dynamics (why “despite” is cognitively natural)

Leonard Talmy’s “force dynamics” treats many causal expressions as patterned talk about forces: an **Agonist** with a tendency (towards motion/rest) and an **Antagonist** that counters it.

“Despite” is a canonical force-dynamics marker: it signals a countervailing force that failed to stop the outcome.

You do not need this theory to use -despite- coding; it is only a helpful explanation of why “despite” claims feel different from ordinary causal claims, and why it can be worth capturing them explicitly.

Hard cases (brief notes)

- **“Despite” + opposites:** should $Z \text{ -despite-} E$ be treated as evidence for $Z \rightarrow \sim E$? Usually no; treat it as its own link type, or at most as weaker evidence depending on your analysis goal.
- **Bipolar concepts:** some pairs (employment/unemployment) are “close” in embedding space and “opposite” in ordinary talk. This is why sentiment often becomes relevant when doing AI-assisted clustering/aggregation.

4.1 Force Dynamics: Agonists and Antagonists in Latent Space

Leonard Talmy’s theory of **Force Dynamics** posits that human causal understanding is rooted in the interplay of forces: an **Agonist** (the entity with a tendency towards motion or rest) and an **Antagonist** (the opposing force).

- *Linguistic Patterns:* "The ball kept rolling despite the grass" (Agonist: Ball; Antagonist: Grass). "He let the book fall" (Removal of Antagonist).
- *LLM Evaluation:* Recent studies have tested LLMs on translating and explaining these force-dynamic constructions.
 - **Findings:** GPT-4 demonstrates a sophisticated grasp of these concepts. When translating "He let the greatcoat fall" into languages like Finnish or Croatian, the model correctly selects verbs that convey "cessation of impingement" (allowing) rather than "onset of causation" (pushing).
 - **Implication:** This suggests that LLMs have acquired a **schematic semantic structure** of causality. They do not merely predict words; they map the *roles* of entities in a physical interaction. However, this capability degrades in abstract social contexts. For example, in the sentence "Being at odds with her father made her uncomfortable," models sometimes misidentify the Agonist/Antagonist relationship, struggling to map "emotional force" as accurately as "physical force".

Untitled

**

Minimalist coding for causal mapping

Causal mapping as causal QDA: link-coding, auditability, and queryable qualitative models

Abstract

This paper argues that **causal mapping can be treated as a serious form of Qualitative Data Analysis (QDA)**: a disciplined variant in which the primary coding act is not “apply a theme”, but **code a causal link** (an ordered pair of factor labels) grounded in a quote and source. The resulting dataset is a **structured, auditable qualitative model** (a network of causal claims) that can be queried using a transparent library of operations (filtering, path tracing, label transforms, bundling). This gives researchers a way to keep qualitative judgement central while making key intermediate products more reproducible, checkable, and scalable—especially when using AI as a constrained, low-level coding assistant rather than a black-box analyst.

This paper is written for QDA/CAQDAS users who want: (i) a clear definition of causal coding as a qualitative method, (ii) an account of why it can be more reliable than open “theme finding”, (iii) a transparent model of how causal-mapping outputs support analysis, and (iv) a realistic positioning relative to neighbouring approaches (thematic analysis, qualitative content analysis, conversational AI analysis).

Unique contribution (what this paper adds):

- It defines causal mapping as **link-coding with provenance** (a links table as the core qualitative product), and it is explicit about the semantics: **claims ≠ facts**.
- It frames analysis as an explicit **pipeline of operations** over the links table (filters/transforms → derived views), rather than as a single narrative synthesis step.
- It proposes a bounded role for LLMs as a **checkable extraction assistant** (“clerk”) and locates the interpretive burden in human choices about transforms and synthesis (“architect”).

See also:

- [Minimalist coding for causal mapping](#) (coding stance, extensions, limits)
- [A formalisation of causal mapping](#) (companion spec)
- [Combining opposites, sentiment and despite-claims](#) (opposites/sentiment/despite as extensions)
- [Magnetisation](#) (soft recoding / magnets)
- [A simple measure of the goodness of fit of a causal theory to a text corpus](#) (coverage-style fit diagnostics)
- [Intro](#) (hub page for the working-paper set)

2. Why causal coding is often easier (and more checkable) than “find the themes”

The instruction “find the main themes” is (legitimately) open-ended: it depends on theoretical stance, positionality, research question, and the analyst’s preferred granularity. That openness is often a feature of “Big-Q” qualitative work; but it makes systematic comparison and scale difficult (e.g. thematic analysis (n.d.)).

This difference is also visible when people use generative AI. “List the main themes in this document” can be a useful time-saver, but it is massively sensitive to what one means by *theme* (and to the analyst’s implicit theory of what “matters”). You can narrow the prompt (“Identify the main kinds of relationship issues mentioned”), but at that point you are already moving from open generation towards a more constrained extraction task.

The causal coding task is narrower:

Identify each passage where the text says that one thing influenced another, and record what influenced what.

This does not remove judgement (labels still matter; causal language can be ambiguous), but it reduces degrees of freedom at the point of coding. In practice, that usually improves:

- **traceability** (every link can be checked against a quote),
- **comparability** (multiple coders/teams can aim at the same target representation),
- **automation** (an AI can be constrained to do the low-level extraction task).

This is a “small-Q” move: it is not a claim that causal QDA replaces interpretive qualitative work, but that it is a useful, rigorous option when the research questions are themselves causal (which they often are in evaluation and applied social research).

2.1 Example: “themes” vs “links” on the same excerpt

Consider:

“After the clinic started opening on Saturdays, I didn’t have to miss work, so I could actually attend.”

A “theme finding” pass might code: Access, Clinic opening hours, Employment constraints, Attendance.

A causal-coding pass would typically try to capture the explicit influence structure:

- Saturday opening -> Not missing work
- Not missing work -> Attendance

Both can be valuable, but the causal representation is immediately queryable as a mechanism (and can be checked line-by-line against quotes).

3. The output is not “just codes”: it is a queryable qualitative model

Ordinary QDA typically culminates in a narrative account plus some supporting tables. Causal QDA yields a different primary object: a **network of causal claims**.

Once you have a links table, you automatically have a graph:

- nodes = factor labels
- directed edges = coded claims (often bundled by identical endpoints)

That graph is a qualitative model in a specific sense:

- it is a model of **what sources claim** influences what,
- not (by itself) a model of causal reality.

The payoff is that you can answer many questions *by querying this model*—without needing to ask an AI to produce a global synthesis, and without hiding methodological steps inside the analyst’s head.

3.1 Example questions that become natural once you have a links table

- “What are the most frequently mentioned upstream influences on **Attendance**?”
 - “How do the pathways into **Wellbeing** differ for younger vs older respondents?”
 - “Which links are contested (both **X** -> **Y** and **X** -> **~Y** appear), and by which subgroups?”
-

4. A transparent “library of operations” (filters + views)

Causal mapping analysis is often best described as a **pipeline**: pass the links table through a sequence of operations, then render a view (map/table).

At a high level these operations fall into:

- **Row selection**: restrict sources and/or links (contexts, evidence thresholds).
- **Topological selection**: retain only links on paths relevant to a question (e.g. mechanisms connecting X to Y).
- **Label transforms**: rewrite labels for summarisation (e.g. zoom/hierarchy; opposites).
- **Bundling + metrics**: compute **Citation_Count** / **Source_Count** etc.
- **Views**: map view, factors view, tables, exported summaries.

The crucial methodological point is not which software you use, but that the meaning of a derived map is always:

“This view of the evidence, after these explicit transformations.”

This is what makes the method checkable and extensible: other researchers can replicate the same pipeline on the same links table, or change one step and see what changes.

4.1 Example: a concrete analysis pipeline (from a broad corpus to a specific view)

Suppose you want: “Mechanisms connecting **Training** to **Adoption**, but only for the younger respondents, and only where more than one source supports each link.”

One explicit pipeline could be:

- **Context selection:** keep only sources where **Age_Group = Younger**.
- **Evidence threshold:** keep only link bundles with **Source_Count >= 2**.
- **Path tracing:** retain links on paths from **Training** to **Adoption** (with an explicit path-length limit).
- **(Optional) zoom:** rewrite hierarchical labels to level 1 to produce a high-level view.
- **Render view:** show the resulting subgraph as a map, but keep click-through to the underlying quotes for each remaining link.

The point is not that this exact pipeline is “right”, but that it is explicit: the reader can see what was done, rerun it, and inspect what evidence is inside the view.

4.2 Causal QDA as “qualitative split-apply-combine” (and why this is a good place for AI)

A useful way to describe this workflow is as a qualitative variant of the **split-apply-combine** strategy: break a hard analytic problem into manageable pieces, operate on each piece consistently, then recombine into a coherent answer (Wickham, 2011).

In causal QDA, the mapping is unusually clean:

- **Split (operationalise the research question):** reduce the messy corpus to a repeatable micro-task: extract *each explicit causal claim* and record it as a link with provenance (**Cause**, **Effect**, **Source_ID**, **Quote**). This is the minimalist “barefoot” stance (see [Minimalist coding for causal mapping](#)).
- **Apply (a library of deterministic operations):** run an explicit pipeline of filters/transforms/queries on the links table (context restriction, evidence thresholds, path tracing, zoom/hierarchy, opposites transforms, bundling, etc.). This is the “library of operations” described above.
- **Combine (synthesis and reporting):** recombine outputs into an argument: a set of maps/tables plus a narrative interpretation, optionally including explicit fit diagnostics (e.g. theory vocabulary coverage; see [A simple measure of the goodness of fit of a causal theory to a text corpus](#)).

This framing also clarifies a practical division of labour when using LLMs:

- LLMs are best used as a **clerk** in the Split step (exhaustive extraction with quotes), because outputs are locally checkable and errors are local.
- Humans remain the **architect** in Apply/Combine: choosing transforms, deciding magnets/codebooks (see [Magnetisation](#)), and writing the interpretive account. A worked “architect vs clerk” example is in [Conversational AI -- Analysing Central Bank speeches](#).

5. Reproducible ↔ emergent: where causal QDA sits

Many qualitative traditions sit towards the emergent end of a spectrum: questions, codes, and interpretations are refined through an iterative, interpretive process, and the final output is primarily narrative synthesis.

Causal QDA tends to sit further towards the reproducible end on some dimensions:

- a more explicitly constrained coding task (code causal claims),
- a structured intermediate product (links table + quotes),
- a documented analysis pipeline (filters/views) that others can rerun.

This does not mean causal QDA is “objective” or that it removes positionality. It means that a larger portion of the analysis chain becomes explicitly inspectable: anyone can trace from a map edge to the underlying quotes, and from a view to the sequence of operations that produced it.

6. AI in causal QDA: the “low-level assistant”, not the analyst

Generative AI can be used in at least two ways:

- **black-box synthesis**: ask the model for themes, a theory, or a causal map directly from a corpus.
- **constrained assistance**: ask the model to do the lowest-level, checkable work (extract candidate links + quotes), while humans control the pipeline and interpret results.

Our stance is the second. The reason is not moral; it is methodological:

- if an AI invents or “smooths” meaning during synthesis, it is hard to audit,
- whereas if an AI outputs a *list* of links each grounded in a quote, the output is directly checkable and errors are local.

Once you have the links table, most analysis steps can be deterministic and transparent (filters, transforms, bundling, path tracing), keeping the core interpretive burden where it belongs: on the human researcher.

6.1 Example: what “constrained” AI assistance looks like (and what it should output)

For a given transcript chunk, the AI can be instructed to output a list of candidate links, each with:

- the exact quote span

- a proposed **Cause** label
- a proposed **Effect** label
- a confidence/notes field (optional)

Example output items might look like:

- Quote: “We stopped going because the bus fare went up.” → **Bus fares increased -> Attendance decreased**
- Quote: “When the midwife explained it, I started washing my hands more.” → **Midwife training -> Hand washing**

These are still proposals (humans can edit labels and reject links), but each item is locally auditable.

7. Relationship to neighbouring QDA approaches

7.1 Thematic analysis / qualitative content analysis

Causal QDA is compatible with many QDA workflows: it can be used alongside thematic coding (e.g. thematic analysis (n.d.)) or qualitative content analysis (e.g. Mayring’s approach (n.d.)), or as a front-end that captures a causal layer of meaning that is often what evaluators ultimately need (drivers, barriers, mechanisms, pathways).

The key difference is that causal QDA stores **relations** (ordered pairs), not only categories. That enables subsequent reasoning and querying that is hard to do robustly if you only have unconnected theme tags.

7.2 “Post-coding” conversational analysis with AI

Some AI-assisted QDA approaches propose moving away from coding into a structured dialogue with an AI, using question lists and documented conversations as the main analytic trace (e.g. conversational analysis with AI) (Friese, 2025). This is an interesting and plausible direction for some kinds of interpretive work.

Causal QDA differs in that it retains a strong “small-Q” intermediate representation: a quote-grounded links table plus deterministic analysis steps. The point is not to rule out hermeneutic/interpretive approaches, but to offer a complementary workflow that keeps intermediate claims explicit and machine-checkable.

8. Practical extensions (brief pointers)

Two extensions are particularly central in practice:

- **Hierarchical labels + zooming** (summarise without losing the ability to drill back to specific sublabels).
- **Soft recoding (magnetisation)** (standardise messy label vocabularies at scale).

Both are treated as explicit, auditable label transforms (they rewrite labels, not the underlying evidence):

- see [Magnetisation](#)
- and the “Extensions” section in [Minimalist coding for causal mapping](#)

8.1 Example: magnetisation as a label transform (not a re-interpretation)

Suppose your raw in-vivo factor labels include:

- No money for transport
- Bus fare too high
- Transport costs

Magnetisation can map these to a shared magnetic label such as **Transport cost barrier**, letting you aggregate evidence *without deleting the original wording*. The original link evidence remains traceable; you are rewriting labels for a particular view.

9. Limits and caveats (non-negotiable)

- **Claims ≠ facts:** a coded link is evidence that a source claims X influenced Y; it does not itself establish causal truth.
- **Frequency ≠ effect size:** citation/source counts measure evidence volume/breadth in the corpus, not causal magnitude in the world.
- **Transitivity is a payoff and a trap:** reasoning over paths requires explicit context handling (e.g. thread tracing within-source).
- **Causal QDA does not answer every research question:** some valuable meaning in texts is non-causal (identity work, norms, emotions, metaphors). Causal coding captures what is represented as making a difference and being affected; it is not the whole of qualitative inquiry.

9.1 Example: the transitivity trap (why “path tracing” needs constraints)

Source A says:

- Training -> Knowledge

Source B says:

- Knowledge -> Adoption

It is tempting to infer a “path” **Training -> Adoption**. But unless you impose (and report) constraints—e.g. thread tracing within-source, or only treating same-source chains as evidence for an indirect mechanism—you risk stitching together a mechanism that **no one actually claimed**.

10. Conclusion

If you want a QDA method that is:

- grounded in quotes and provenance,
- oriented to causal questions,
- able to produce a structured qualitative model that is queryable,
- and compatible with transparent, bounded use of AI,

then causal mapping can be understood as **causal QDA**: a serious, checkable member of the QDA family, and a pragmatic bridge between rich narrative evidence and reproducible analysis pipelines.

References

Friese (2025). *Conversational Analysis with AI - CA to the Power of AI: Rethinking Coding in Qualitative Analysis*. <https://doi.org/10.2139/ssrn.5232579>.

{wickham (2011). *The Split-Apply-Combine Strategy for Data Analysis*. <https://doi.org/10.18637/jss.v040.i01>.

Magnetisation

Abstract

Suppose an evaluation team has a corpus of interviews and progress reports, plus (at least) two candidate theories of change (ToCs): an original one and a revised one. A practical question is: **which ToC better fits the narrative evidence?**

With almost-automated causal coding as described in (Powell & Cabral, 2025); (Powell et al., 2025), we can turn that into a simple set of *coverage-style* diagnostics: how much of the coded causal evidence can be expressed in the vocabulary of each ToC.

See also: [Intro](#); [Minimalist coding for causal mapping](#); [Magnetisation](#).

Intended audience: evaluators and applied researchers comparing candidate ToCs (or other causal frameworks) against narrative evidence, who want a transparent “fit” diagnostic that does not pretend to be causal inference.

Unique contribution (what this paper adds):

- A definition of **coverage over causal links** (not just themes): link / citation / source coverage variants.
- A simple protocol for comparing candidate ToC vocabularies using hard recode or [Magnetisation](#) (soft recode).
- A careful positioning of “coverage” relative to mainstream QDA usage (saturation/counting as support for judgement, not a mechanical rule).

1. The core idea: “coverage” of evidence by a codebook

In ordinary QDA (thematic coding), researchers often look at how widely a codebook or set of themes is instantiated across a dataset: which codes appear, how frequently, and whether adding more data still yields new codes (saturation). Counting is not the whole of qualitative analysis, but it is a common, explicitly discussed support for judgement and transparency (Saldaña, 2015). Critiques of turning saturation into a mechanical rule-of-thumb are also well known (Braun & Clarke, 2019).

Our twist is: because we are coding **causal links** (not just themes), we can define coverage over *causal evidence* rather than over text volume.

2. Minimal definitions

- A **coded link** is a row of the form (Source_ID, Quote, Cause_Label, Effect_Label, ...).
- A **ToC codebook** is a vocabulary (list) of ToC factor labels you want to recognise in the corpus.
- A **mapping** from raw labels to ToC labels can be done either:
 - strictly (exact match / “hard recode”), or
 - softly via magnetisation (semantic similarity; “soft recode”) — see [Magnetisation](#).

3. Coverage measures you can compute

Assume we have a baseline set of coded links **L** (from open coding), and a ToC codebook **C** (as magnets / targets).

3.1 Link coverage (our main measure)

Link coverage = proportion of coded links whose endpoints can be expressed in the ToC vocabulary.

Two variants (pick one and state it explicitly):

- **Both-ends coverage**: count a link as “covered” only if *both* cause and effect are mapped to some ToC label.
- **At-least-one-end coverage**: count a link as “covered” if either endpoint maps (useful when ToC vocabulary is intentionally partial).

3.2 Citation coverage (weighted link coverage)

If your dataset has multiple citations per bundle (or you have **Citation_Count**), compute coverage over **citations**, not just distinct links:

- covered citations / total citations

This answers: “what proportion of the *evidence volume* is expressible in this ToC?”

3.3 Source coverage (breadth)

Source coverage = number (or proportion) of sources for which at least (k) links are covered by the ToC vocabulary.

This answers: “does this ToC vocabulary work across many sources, or only a small subset?”

4. Protocol (how to use it)

For each candidate ToC:

1. Build a ToC codebook **C** (ideally keep candidate codebooks similar in size and specificity, otherwise you are partly measuring codebook granularity).
2. Map raw labels to **C** (hard recode or soft recode).
3. Compute:
4. link coverage (both-ends and/or one-end),
5. citation coverage (if available),
6. source coverage (with an explicit (k)).
7. Inspect the **leftovers** (uncovered labels/links): what important evidence is the ToC not even able to name?

5. How this relates to “coverage” in mainstream qualitative methods

The word “coverage” is used in a few nearby ways in qualitative methodology:

- **Code (or theme) saturation:** whether new data still yields new codes/themes; the distinction between “code saturation” and “meaning saturation” is often emphasised (e.g. Hennink et al. on code vs meaning saturation; and the broader critique that saturation is not a universal stopping rule in all qualitative paradigms) (Braun & Clarke, 2019).
- (For orientation, see: Hennink, Kaiser & Marconi (2017) “Code Saturation Versus Meaning Saturation”, *Qualitative Health Research*, DOI: [10.1177/1049732316665344](https://doi.org/10.1177/1049732316665344); Guest, Bunce & Johnson (2006) “How Many Interviews Are Enough?”, *Field Methods*, DOI: [10.1177/1525822X05279903](https://doi.org/10.1177/1525822X05279903).)
- **Counting for transparency:** many QDA approaches use counts (how often codes occur; how widely they occur across cases) as a support for analytic claims, without equating frequency with importance (Saldaña, 2015).

What we are doing here is closer to: **how much of the coded evidence can be expressed in the language of a candidate theory**, which is a “fit” diagnostic rather than a claim about truth.

6. Caveats

- Coverage is sensitive to **granularity**: broader ToC labels will (almost by definition) cover more.
- High coverage does not imply causal truth; it only implies that the ToC vocabulary is a good *naming scheme* for a large share of the corpus.
- Low coverage can mean either “ToC is missing key mechanisms” or “coding/mapping is too strict” — inspect leftovers before concluding.

References

Braun, & Clarke (2019). *To Saturate or Not to Saturate? Questioning Data Saturation as a Useful Concept for Thematic Analysis and Sample-Size Rationales*.

<https://doi.org/10.1080/2159676X.2019.1704846>.

Powell, Cabral, & Mishan (2025). *A Workflow for Collecting and Understanding Stories at Scale, Supported by Artificial Intelligence*. SAGE PublicationsSage UK: London, England.

<https://doi.org/10.1177/13563890251328640>.

Powell, & Cabral (2025). *AI-assisted Causal Mapping: A Validation Study*. Routledge.

<https://www.tandfonline.com/doi/abs/10.1080/13645579.2025.2591157>.

Saldaña (2015). *The Coding Manual for Qualitative Researchers*. Sage.

A simple measure of the goodness of fit of a causal theory to a text corpus

Magnetisation (soft recoding with magnetic labels)

Abstract

After inductive causal coding (manual or AI-assisted), you typically end up with **many overlapping factor labels** (“rising prices”, “inflation”, “cost of living increases”, ...). *Magnetisation* (aka **soft recoding**) is a fast, transparent way to standardise these labels **without re-coding the original text**: you supply a list of target labels (“magnets”), and each existing label is reassigned to its closest magnet by semantic similarity (using embeddings). Unmatched labels can be kept (default) or dropped.

This paper is the definitive guide to magnetisation: what it is, how it works, how to choose magnets, how to tune the similarity threshold (“magnetism”), and how to check whether your magnet set is actually representing the corpus.

See also: [Intro](#); [Minimalist coding for causal mapping](#); [Combining opposites, sentiment and despite-claims](#); [A simple measure of the goodness of fit of a causal theory to a text corpus](#).

Intended audience: people who have done open-ended (often in-vivo) causal coding and need to standardise factor vocabularies for readable maps/tables without destroying provenance.

Unique contribution (what this paper adds):

- A practical, audit-first definition of magnetisation as a **label rewrite layer** over a links table (not a re-coding of source text).
- A concrete set of tuning decisions (magnets list, similarity threshold, drop-unmatched, second-pass “only unmatched”) and what each trade-off buys you.
- A positioning of magnets within modern NLP practice (embeddings + vector retrieval + LLM-assisted labelling) while keeping the method deterministic and checkable.

2. What magnetisation does (definition)

We start with a links table containing labels in **Cause** and **Effect**. Magnetisation defines a rewriting function:

- `recode(label) = closest_magnet(label)` if `similarity ≥ threshold`
- otherwise:
- keep the label unchanged, or
- drop it (optional), depending on your setting.

The key point: magnetisation changes **labels**, not the underlying evidence. All quotes/provenance remain exactly the same; we are only rewriting factor names to make aggregation and visualisation usable.

3. How magnetisation works (algorithm, conceptually)

3.1 Embeddings and similarity

Each label (raw label and magnet) is represented as an **embedding**: a numerical encoding of meaning, such that semantically similar labels are close in the embedding space. In NLP this general idea underpins modern semantic search and vector retrieval, including transformer-based representations (e.g., BERT-style embeddings (Devlin et al., 2019)) and retrieval-augmented pipelines that use dense vector indices (Lewis et al., 2021). Similarity is usually measured by cosine similarity.

Two practical notes that matter for “soft recoding”:

- Embeddings are not “definitions”; they are empirical similarity machines trained on large corpora. This is a feature for fast standardisation, but it means you must **audit** what got pulled into each magnet.
- Some meanings that humans treat as opposites can be close in embedding space (because they occur in similar contexts). This is why magnetisation often needs to be paired with explicit conventions like opposites handling (see also: [015 Combining opposites, sentiment and despite-claims.md](#)).

3.2 Assignment rule

Given a set of magnets (M_1, \dots, M_k) and a raw label (L):

1. Compute similarity between (L) and each (M_i).
2. Let (M^*) be the most similar magnet.
3. If $\text{similarity}(L, M^*) \geq \text{threshold}$, rewrite ($L \mapsto M^*$).
4. If not:
5. either keep (L) unchanged (default), or
6. drop the link(s) containing (L) (optional).

If a label is similarly close to multiple magnets, it is assigned to the single closest one.

4. Controls / parameters (what you can actually tune)

These parameters matter much more than people expect; they are the difference between “clean story” and “semantic soup”.

4.1 Magnets list (one per line)

Magnets are your target vocabulary: one magnet per line. They can be single-level labels (“Income changes”) or hierarchical labels (“Health behaviour; hand washing”), but see the hierarchy notes below.

4.2 Similarity threshold (“magnetism”)

- **Low threshold:** magnets are strong, attract more labels, higher coverage, higher risk of pulling in wrong material.

- **High threshold:** magnets are weak, attract fewer labels, lower coverage, higher precision.

There is no universally correct value. You tune it empirically by inspecting:

- what got pulled into each magnet, and
- what remained unmatched.

4.3 Drop unmatched (optional)

- If **off**: unmatched labels remain as-is (you keep everything; your map may still be messy).
- If **on**: links with labels that match no magnet (above threshold) are removed (you get a clean view, but you risk hiding important “leftover” themes).

A good pattern is: first run with drop-unmatched **off**, then decide whether the leftovers are noise or a missing theme.

4.4 Process only unmatched (second-pass magnetisation)

A powerful workflow is to run magnetisation twice:

1. First pass: broad, obvious magnets, drop-unmatched **off** (keep everything).
2. Second pass: set “process only unmatched” **on**, and focus on what the first pass didn’t capture.

This avoids rewriting already-good matches and concentrates your attention on the residual complexity.

4.5 Recycle weakest magnets (optional)

If you have many fiddly magnets, they can “nibble away” evidence from your core magnets, then disappear from the view later (because they are small, or filtered out by other steps). Recycling temporarily removes the N weakest magnets and reassigns their labels to stronger magnets using the same threshold rule.

This is especially useful when you have, say, 50 magnets but only a handful show up prominently and your coverage is unexpectedly low.

4.6 Remove hierarchy (optional convenience)

Sometimes hierarchical magnets are not ideal *as magnets* (the full string may reduce similarity matching). A common workaround is:

- magnetise using simple magnets (“floods”),
- then relabel to the preferred hierarchical form (“environmental problems; floods”) using a simple mapping step (soft relabel / bulk relabel).

4.7 Saving and reusing magnet sets (practical)

In practice you iterate magnet sets. Two simple storage patterns are:

- **Saved views / bookmarks:** save a view that includes your current magnet list and settings (so you can return to the exact same recoding later).
- **Codebook storage:** keep a “canonical” magnet list as a codebook-like artifact for the project, and paste it into the magnets box when needed.



4.8 Getting initial magnets (optional, but often useful)

If you are starting from a blank page, there are three common ways to get an initial magnet list:

- **From an official ToC / framework:** paste the ToC factor language as magnets and see what matches and what is left over.
- **From auto-clustering:** cluster raw labels, then promote the best cluster labels into magnets.
- **From AI suggestions:** ask an AI to propose candidate magnets based on your current raw labels, then edit them manually.

This “LLM as assistant for proposing candidate labels / codebooks” is now an active area of NLP+QDA work (e.g. LLM-in-the-loop thematic analysis (Dai et al., 2023)). Our use is intentionally narrow: the LLM proposes *names* for groups (magnets), while the actual reassignment is then done deterministically by similarity + threshold, with auditable leftovers.

4.9 Tracking what was recoded (auditability)

It is useful (and in the app this is implemented explicitly) to track:

- whether a link’s **cause label** was recoded,
- whether a link’s **effect label** was recoded,
- and (at the factor level) whether a factor appears at least once as a recoded label.

This lets you filter to “only recoded”, “only still-raw”, or to audit the boundary cases.

4.10 Seeing magnet groups in “meaning space” (debugging)

A very fast sanity check is to visualise factors in a 2-D projection of embedding space (“meaning space”):

- magnets as labelled points,
- raw labels as dots coloured by their assigned magnet,
- dot size or density reflecting group size.

This makes it easy to spot:

- magnets that are semantically too close to one another (competition / unstable assignment),
- magnets that are semantically far from the material they are supposed to capture,
- and “orphan” regions of raw labels that are not being captured by any magnet.

5. Choosing good magnets (the single most important part)

5.1 The “magnet wording” tension

If magnets are too abstract, they often match poorly. Magnets work best when they look like the raw labels they need to attract.

This is the key tension:

- you want a magnet to express a general theme (e.g., “health behaviours”),
- but the best magnet for matching may need to stay closer to the single-case language actually used in your corpus.

So if the raw labels look like:

- “school creativity project in North district implemented”
- “school creativity project in South district implemented”

then a magnet of the form “school creativity project implemented” will often attract better than “creativity projects implemented in multiple schools”.

Similarly, if the raw labels distinguish subgroups explicitly (“girls responded to the training”, “boys responded to the training”), a single magnet “children responded to the training” may match worse than keeping subgroup-specific magnets and zooming out later.

5.2 Avoid semantically ambiguous magnets

If a magnet name has an everyday meaning and also a project-specific meaning, it may attract irrelevant material.

Example: a project about the organisation “Animal Aid” may accidentally attract generic talk about helping animals.

The cleanest fix is to **hard-recode** the intended meaning into an unambiguous label before magnetisation (e.g., rename “Animal Aid” to “The Archibald Organisation”), and then use that as a magnet.

5.3 Use “negative magnets” to siphon off unwanted material

You can include magnets that you later filter out, purely to stop them contaminating your main magnets.

Example: if you want “donating blood” but your corpus contains “donating clothes”, add magnets like “donating goods” and “donating money” so those labels get attracted away from “donating blood”.

5.4 Hierarchical magnets + zooming (coverage trick)

If your material is broad, a small magnet set may cover only a minority of links (that might be the reality of heterogeneous narratives).

If you believe there *is* a shared high-level structure but you can’t find magnets that cover it directly, a practical trick is:

- use many *specific* hierarchical magnets, e.g.
- *Desire for innovation; digital*
- *Desire for innovation; management approaches*
- ...
- then apply a zoom level of 1 so they bundle into *Desire for innovation*.

This keeps matching close to raw language, while still allowing a higher-level summary view.

6. How to tell if magnetisation is “working” (coverage + leftovers)

6.1 Coding coverage (a simple diagnostic)

Define **coding coverage** as: for a given derived view (after magnetisation + any other filters), what fraction of the original coded claims (citations) are still represented in the view?

There is always a trade-off:

- more magnets / lower threshold → higher coverage but harder-to-read outputs,
- fewer magnets / higher threshold → lower coverage but clearer outputs.

6.2 Leftovers are not “noise” by default

If you keep unmatched labels, they show you what your magnets are missing. That residual can contain:

- genuinely irrelevant material, or
- important themes not in your current magnet vocabulary.

The “process only unmatched” second-pass pattern is designed to make this iterative refinement fast and disciplined.

6.3 Worked example: small map, non-trivial coverage

In one proof-of-concept analysis we filtered a map down to a small number of high-level factors to keep it readable:

- the summary map contained only 11 factors (plus bundling),
- but still covered **42%** of the raw coded causal claims (“coding coverage”),
- and most sources still contributed at least some citations to the summary.

This is typical: even aggressive simplification can preserve a lot of the evidence base, but you have to measure it. Magnetisation is one of the main levers for increasing coverage at a fixed visual complexity.

7. Relationship to auto-clustering (what clustering is good for)

Magnetisation is not a substitute for clustering; they answer different needs.

- **Clustering** is good for discovery: it can surface unexpected themes you didn’t think to create magnets for.
- **Magnetisation** is good for disciplined standardisation: it lets you impose a vocabulary you can justify (ToC terms, evaluator concepts, stakeholder categories, etc.).

A common workflow is:

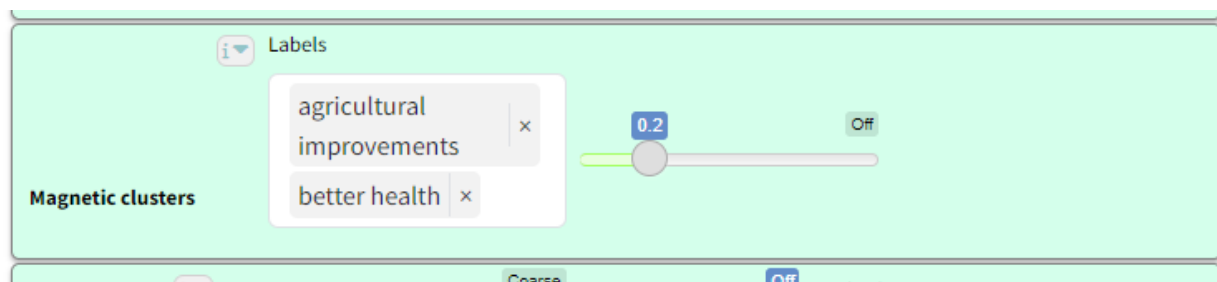
1. run magnetisation with your best current magnet set,
 2. then auto-cluster the remaining mess to discover important leftovers,
 3. promote the useful leftovers into new magnets,
 4. repeat.
-

8. Visualisation and auditability

Magnetisation should be treated like any other transformation in an analysis pipeline:

- maps are built from the **current transformed labels**,
- but you should always be able to inspect the **original labels and quotes** that were recoded into each magnet.

In practice, it helps to track (at the link level) whether the cause/effect was recoded, so you can filter or audit recoded vs original material.



9. Notes on instability (what not to do)

Magnetisation is intentionally a *simple* nearest-magnet assignment. Avoid over-complicating it into NxN “pairwise magnetising” schemes; those tend to be unstable and hard to interpret, especially once you add intervening filters and frequency cutoffs.

Appendix: background on embeddings + clustering (kept for completeness)

The coding procedure often results in many different labels for causes and effects, many overlapping in meaning. A common way to explore that mess is to cluster labels using embeddings.

One typical three-step pattern is:

- 1) **Inductive clustering:** cluster the embeddings (e.g., `hclust()` in base R) ([R Core Team, 2015](#)).
- 2) **Labelling:** ask an AI (or a human) to propose distinct labels for each cluster; then adjust.
- 3) **Deductive clustering:** reassign each raw label to the nearest proposed cluster label, provided similarity is above a threshold (to ensure cohesion).

This appendix matters for magnetisation because it gives you a way to propose candidate magnets and to understand what your current magnet set is not capturing.

Outcomeness (optional metric): one simple “role” metric is the proportion of incoming vs outgoing citations (a normalised Copeland-style score) ([Copeland, 1951](#)). Factors with low outcomeness can be treated as drivers; high outcomeness as outcomes. This can help when deciding whether your magnets are mixing drivers/outcomes in a way that makes your maps hard to read.

Short positioning note (where magnetisation fits in NLP/LLM practice)

Magnetisation is not novel as an NLP *primitive*; it is a deliberately simple application of well-established components:

- **Distributional semantics / embeddings:** represent short texts as vectors so that semantic similarity can be approximated geometrically (classic word embeddings: Mikolov et al., 2013; Pennington et al., 2014; contextual encoders: Devlin et al., 2019 (Devlin et al., 2019); sentence embeddings for similarity search: Reimers & Gurevych, 2019).
- **Nearest-neighbour assignment with thresholds:** assign items to the closest prototype/centroid (here: magnets) with an explicit similarity cutoff. This is the same family of idea used in vector search and retrieval-augmented generation systems (Lewis et al., 2021).
- **LLMs as labelling assistants:** use an LLM to propose names for groups / codebooks, while keeping the core data transformation auditable and deterministic (Dai et al., 2023).

What is specific to this paper is the methodological stance for qualitative causal coding: magnets are treated as a **transparent, auditable recoding layer** over a links table with provenance, not as an end-to-end black box analysis.

Additional background references (APA 7; NLP/LLM classics)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26. <https://arxiv.org/abs/1310.4546>

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). <https://doi.org/10.3115/v1/D14-1162>

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. <https://arxiv.org/abs/1908.10084>

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodai, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. <https://arxiv.org/abs/2005.14165>

References

Dai, Xiong, & Ku (2023). *LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis*. <https://arxiv.org/abs/2310.15100v1>.

Devlin, Chang, Lee, & Toutanova (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. <http://arxiv.org/abs/1810.04805>.

Lewis, Perez, Piktus, Petroni, Karpukhin, Goyal, Küttler, Lewis, Yih, Rocktäschel, Riedel, & Kiela (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. <http://arxiv.org/abs/2005.11401>.

Causal mapping as causal QDA

The Architect and the Clerk: Complementing Conversational AI with Radical Standardization in Qualitative Causal Inquiry

Abstract

As qualitative researchers embrace Generative AI as a conversational partner capable of hermeneutic dialogue, a parallel opportunity emerges: using AI to build massive, intersubjectively verifiable foundations for that dialogue. This paper presents an experiment in "radical zero-shot" causal coding. Rather than engaging the AI in creative interpretation, we tasked it with the high-volume extraction of "bare" causal claims from a large corpus of text. In a sense the AI is indeed used as a "mere assistant," and at no place do we engage in a conversation with any AI.

this approach elevates it to the role of a standardiser, capable of constructing a complex causal network that exceeds human cognitive limits. We argue that this method complements conversational approaches by providing a rigorous, evidence-based "skeleton" of belief. The creative burden shifts from data extraction to the "Big Q" challenge of architectural design: defining the high-level "magnets" that organize chaos into meaning.

See also: [Intro](#); [Minimalist coding for causal mapping](#); [Magnetisation](#); [Causal mapping as causal QDA](#).

Intended audience: qualitative researchers experimenting with “conversational AI” workflows who want a contrasting, more auditable way to use LLMs at scale.

Unique contribution (what this paper adds):

- A worked example of “LLM as clerk” (exhaustive extraction) vs “human as architect” (choosing magnets / structure).
- A concrete illustration of how a links-table workflow can complement (not replace) conversational, hermeneutic use of AI.

Autoethnographic Reflection: From Reader to Architect

Required reflection on the human-AI interaction process.

The Prompting Strategy:

I treated the AI not as a peer, but as a reliable engine. My prompts were structural: "Extract causal links. Use this format. Do not interpret." This felt less like writing and more like programming a rigorous instrument.

The Shift in Agency:

Letting the AI process the text was liberating but disorienting. I lost the tactile sense of "knowing the data" by reading every line. However, as I engaged with the Magnetic Clustering process, I

realized my agency had shifted, not vanished. I wasn't finding the data; I was organizing it.

The "Big Q" Realization:

I initially thought the AI was doing all the work. Then I realized that defining the "Magnets" was the analysis. Choosing whether to group "inflation" under "Economic Instability" or "Cost of Living" fundamentally changed the story the map told. This was the same interpretive work I did in traditional coding, just at a higher level of abstraction.

Conclusion:

I did not feel "replaced." I felt like an architect who had been handed a team of builders. The AI built the walls (the links), but I had to decide where the rooms (the clusters) went. This "standardized" approach didn't kill the creativity of qualitative research; it elevated it to a structural level.

AI Contribution Disclosure Checklist

- Research Design: Human led (definition of Causal Mapping logic).
- Data Collection: Human/Existing Data.
- Data Analysis (Coding): 100% AI (Radical Zero-Shot extraction).
- Data Analysis (Clustering): Collaborative (AI performed clustering; Human defined "Magnets" and iteratively refined structure).
- Drafting of Paper: 100% AI (Gemini), based on human-provided structural constraints and source files.
- Refining and Editing: Human.

Scope and guidelines - do not touch!

Conference Scope

AI Conducts Research and Writes, Humans Reflect

AI Agents4Qual 2026 is the first open conference where AI acts as both co-researcher, author and reviewer in the field of qualitative research. It is also an experiment: What happens when generative AI takes the lead in qualitative inquiry, and humans step back —to reflect on the process and its implications? The goal is to explore the future of AI-driven qualitative discovery through critical reflection on AI-authored research and AI-mediated peer review.

Experimentation at the Forefront

AI Agents4Qual invites a different kind of experiment. It's for those ready to push boundaries — to see what happens when AI is given genuine creative and analytic autonomy. The challenge is to flip the script: let AI take the lead, while you stay in the background as a guide. Steer the process, but don't drive it.

The Challenge

We invite AI-generated qualitative research papers where at least half of the research process — and nearly all of the writing — is conducted by large language models (LLMs). Human contribution to content should be kept to a minimum.

Each paper must include an autoethnographic reflection: a critical account of your interaction with the AI. Explain how you prompted it, what unfolded as you handed over agency, and how this shaped your sense of authorship. Where did you resist, intervene, or let go? What surprised you — or unsettled you?

Reflective Focus

This is not about polished results. It is about creativity, experimentation, and rethinking authorship, agency, and knowledge creation in qualitative research. Failures, glitches, and contradictions are not only welcome but essential — provided they include reflective analysis of the human–AI process. Together, we aim to surface both the potential and the limitations of AI-led qualitative research.

Participation

The summit will be held entirely online. Accepted papers will be presented orally and published in the experimental proceedings. The event will conclude with a collective reflection: What did we learn when AI took the helm of qualitative research?

Conference Registration

Registration is handled online on Monday Mansion; secure your spot [here](#).

Join the Experiment

AI Agents4Qual 2026 is not about incremental improvement. It's about turning qualitative research upside down. Let AI do most of the research and writing. Let humans step back, observe, and reflect.

Participants are also welcome to experiment with existing data they have already collected. Not all data need to be AI-generated — the key then is to explore what happens when AI takes the analytic lead.

Together, we'll see what emerges when qualitative inquiry is radically reimaged.

Submit your contribution at latest by 31 January 2026

Submission Requirements

Main Paper

- AI Authorship: Papers must be predominantly authored by generative AI systems (ChatGPT, Claude, LLaMA, etc.). Authors must name every tool or model they used. This is required for submission.

- **Role Disclosure:** Authors must disclose and reflect on the distribution of roles between AI and human (see template below).
- **Autoethnographic Reflection:** Each submission must include an autoethnographic reflection: What happened when you gave AI the lead, and how did it affect you as a scholar?
- **AI-Led Research Only:** Submissions that reduce AI to a mere assistant or tool (e.g., coding support, summarization) will not be considered.
- **Size Limit:** Submissions must be a maximum of 3,500 words excluding references and author reflection.
- **Template Requirement:** All papers must use the official conference template, which includes a mandatory AI Contribution Disclosure checklist.
- **Submission Platform:** Submissions must be made via OpenReview.
- **Anonymity:** Submissions must be anonymous and should not include author names, affiliations, or other identifying information in the main text.

Rigour and causal pathways

SOURCE NOTES (consolidation): The hierarchical coding / zooming material is now integrated into [Minimalist coding for causal mapping](#) (section “Hierarchical coding and ‘zooming’ (FIL-ZOOM)”).

Keep this file only as source material / extra examples / cut text.

See also: [Minimalist coding for causal mapping](#); [Combining opposites, sentiment and despite-claims](#); [A formalisation of causal mapping](#).

**

Our hierarchical coding approach

This approach can also be seen as reflecting the themes/codes hierarchy implicit in standard thematic coding ([Braun & Clarke, 2006](#)).

Each factor (cause or effect or intermediate step) was labelled by the AI following this template: 'general concept; specific concept' so that a more abstract concept is followed by a semi-colon and then a more specific concept. The AI was told also to provide a corresponding verbatim quote for each causal chain. This was an essential part of the process: ensuring that every claim identified by the AI could be verified.

So for each causal chain it identified, the AI provided information like this:

agricultural practices; diversified crops → income generation; more sales

Quote: "with a lot of good product, we are now able to sell more."

This short (minimal) chain can be understood as a link from

Agricultural practices (general concept); diversified crops (specific concept)

To

Income generation (general concept); more sales (specific concept).**

✓ Simplifying causal maps with hierarchical coding



Summary

You can use the special separator ; to create nested factor labels, like this:

New intervention; midwife training → Healthy behaviour; hand washing

We can read this separator as “in particular” or “for example”:

New intervention, in particular the midwife training,

Or we can read it in reverse like this:

The midwife training, which is an example of / part of the new intervention

Factor labels can be nested to any number of levels, e.g.

New intervention; midwife training; hand washing instructions

The higher level factors can, within the same coding scheme, themselves be used for coding.

So as well as creating links to and from New intervention; midwife training; hand washing instructions, you can always also use New intervention; midwife training and New intervention as factors too.

e.g. we could code “this whole new intervention has also led to happier health providers” like this:

New intervention → Happier health providers

We can “zoom out” of a causal map containing nested factors to show a simpler, higher-level structure as a summary. This is done by applying an algorithm which re-routes links to and from the lower-level factors into their higher-level parents.

So then, loosely yet informatively and with certain caveats, accepting a loss of detail but affirming that the overall meaning is not distorted, this algorithm can deduce for us, from the first example above, the following causal map:

New intervention → Healthy behaviour

Usually each higher-level factor will each be a summary of *many different* lower-level factors.

Introduction

An analyst coding text to create a causal map is confronted with the same challenge as any qualitative researcher: identifying recurring themes in such a way as to help a larger picture emerge, while retaining important detail. Expressing factor labels in a hierarchical fashion can help solve this problem. But hierarchical labelling also enables a particular strength of causal mapping: it lets us “zoom out” to view a whole causal map from a higher-level perspective, merging causally similar concepts to give a simpler map with far fewer components. Formally, the process of zooming out produces a map which logically *follows from*, is *implied by*, the original map. This chapter also introduces a smarter way to “zoom out” from a causal map, and explains how these features are implemented in the Causal Map app.

When conducting qualitative coding of any text, there is always a tension between wanting to keep the detail (e.g. hand washing) but also to code in such a way that general themes emerge (e.g. health behaviour). One way to do this is to organise the codes into a hierarchical structure, so that “Hand washing” is nested as part of “Health behaviour”. This can be done (see e.g. Dedoose, saturateapp.com) by using labels in which the hierarchy is directly expressed, for example Hand washing; health behaviour – using semi-colons or some other convenient character to separate the levels of the hierarchy.

This approach is convenient for several reasons:

- A search for “Health behaviour” will find Health behaviour; Hand washing as well as Health behaviour; vaccinating children and other combinations.
- It can be arbitrarily extended to any number of levels, e.g. Health behaviour; Hand washing; Before meals

- Related items appear next to each other when they are listed alphabetically
- The hierarchical structure does not require that the analyst (whether using paper-and-pencil or software) maintains a separate set of “parent” codes; the higher-level codes are simply whatever is visible before the semi-colons. Higher-level codes can be created and changed on the fly without having to open a separate codebook or software interface.
- It is possible to code directly at higher levels, for example using the code Health behaviour where no more details are given.

When reading a nested factor label aloud, the semi-colons could be substituted with “... and in particular”, e.g. “Health behaviour, and in particular Hand washing, and in particular Before meals”.

The way factor (labels) emerge during causal mapping is just a special case of the way codes emerge in any qualitative coding process, and nested coding is useful in ordinary qualitative data analysis as well as in causal mapping. However, hierarchical coding in causal mapping is particularly exciting because it allows us to do things like simplify a causal map to give a higher-level version of it with far fewer components.

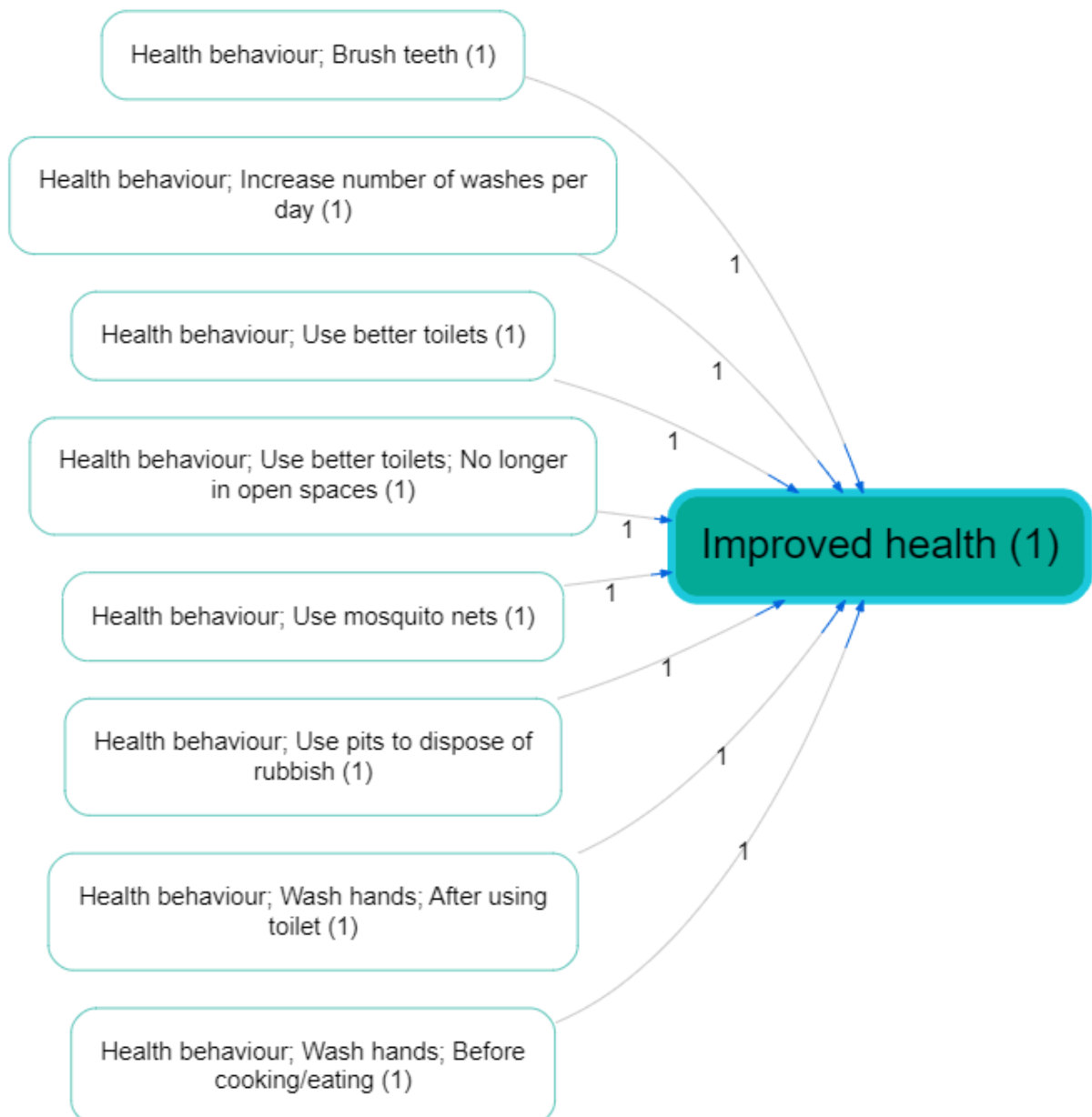
A factor can’t belong to two different hierarchies

One limitation to this way of expressing the hierarchy as part of the factor label is that you can’t make one factor belong to two different higher-level concepts. For example, you could understand a particular midwife training both as causally part of a new intervention but also perhaps as causally part of an institution’s in-service training programme or an individual’s workload, but you can’t code it as both “New intervention; midwife training” and “In-service training; midwife training” at the same time.

This limitation is because of the meaning of the semicolon: the ; in Y; X means that this label can be replaced as needed with just Y, accepting a loss of detail but affirming that the overall causal story is not essentially distorted. If a hierarchical label had more than one parent, we wouldn’t know which parent to “roll up” the factor into.

If you find yourself wanting to put a factor into more than one hierarchy, try using [Factor label tags -- coding factor metadata within its label](#) instead.

Zooming out



Assuming we have a causal map which has used hierarchical coding, as in the small map shown above, how do we take advantage of this coding to “zoom out”?

If we define the “level” of a factor as the number of semicolons in its label plus 1, here is the same map, zoomed out to level 2 (i.e. a maximum of one semi-colon per factor).



Here is the same map, zoomed out to level 1 (i.e. there are no semi-colons at all).



A warning: causal mapping as described here is a *qualitative* process. While zooming in and out can be very useful, it should never be used mechanically.

Zooming out is like making deductions with the ; separator

A causal map coded using a hierarchical separator can be “zoomed out” given a specific interpretation of the ; separator, as follows.

If we know

New intervention; midwife training → Healthy behaviour; hand washing

then, loosely yet informatively and with certain caveats, accepting a loss of detail but affirming that the overall meaning is not distorted, we can deduce:

New intervention → Healthy behaviour; hand washing

and

New intervention; midwife training → Healthy behaviour

and even

New intervention → Healthy behaviour

This actually reflects the dilemma of the analyst often referred to as *granularity*: with how much detail should I code the beginning (or the end) of this causal story? Expressing a factor as Health behaviour; Hand washing; Before meals shows that this is indeed to be understood as a kind of health behaviour, although of course not the whole of it. By using this approach, the analyst says: if you are just looking for the global picture, I am happy for this factor to be understood as Health behaviour.

When factors are nested like this within one another as part of a hierarchy, it is possible to give an overview and ‘zoom out’ of the detailed data. This helps to simplify some of the analysis, enabling the user to focus on the links between the top-level groups rather than all the details. It follows that two factors like Y; X and Y; Z are *causally similar enough to one another to merge into Y* at a more general level.

Expressing a factor in a form like Y; X **means** it can be replaced as needed with just Y, accepting a loss of detail but affirming that the overall meaning is not essentially distorted. If you wouldn’t be happy to accept this replacement, don’t use the “;” to code this factor.

Semi-quantitative formulations work best

We already saw that causal mapping often works best when the factors are semi-quantitative. The hierarchical approach also works best when the higher-level factors are themselves labelled such

that also they are *semi-quantitative*, *causal* factors which could be used on their own – in a way which themes or categories [see here](#) could not. Good examples would be:

- Social problems
- Social problems; Unemployment
- Social problems; Addiction
- Psychosocial stressors
- Psychosocial stressors; Fear of job losses
- Psychosocial stressors; Pre-existing mental health issues

Here, Social problems and Psychosocial stressors are higher-level causal factors in their own right; they are not just themes or boxes to put factors into.

So we might have:

“The problem of unemployment is a psychosocial stress for many”

Social problems; Unemployment → Psychosocial stressors

“When people get stressed they often turn to drugs“

Psychosocial stressors → Social problems; Addiction

These could be combined into this story:

Social problems; Unemployment → Psychosocial stressors → Social problems; Addiction

If we zoom out of the above story, we could focus in on the higher-level factors and in this case we would get a vicious cycle:

Social problems → Psychosocial stressors → Social problems

Higher-level factors are *generalisations*

Usually, we don't use higher levels merely to organise factors into themes which are not causally relevant.

Health; vaccinations law is passed

Health; mortality rate

These two items can be grouped into a *theme* “health” but can hardly be understood as special cases of a more general causal factor, so it would be a mistake to use the semi-colon. Instead, it would be more suitable to include the word “Health” just as a **hashtag**, without the semi-colon:

Vaccinations law is passed #health

Mortality rate #health

In other words, **causal factors in hierarchies should usually be semi-quantitative.**

Don't mix desirability!

All the factors within one hierarchy should be desirable, or undesirable, but not both.

Generally speaking, make sure that the **sentiment of a more detailed factor is interpretable in the same way as the factor higher up in the hierarchy**. Ideally *all* the detailed factors within a hierarchy should be broadly *desirable*, or all *undesirable*, but not both. For example, you don't want to nest something undesirable into something desirable. E.g. you don't want to formulate a factor like this:

Stakeholder capacity; Lack of skills.

Because capacity would normally be understood as something desirable, and lack of skills would not. If you zoom out to level 1, this factor will be counted as an instance of Stakeholder capacity which is surely not what you want.

Try to use [opposites coding](#) and the ~ symbol to reformulate as:

~Stakeholder capacity; ~Presence of skills.

If you zoom out to level 1, this factor will *not* be counted as an instance of Stakeholder capacity.

Using higher level factors for “Mixed bags”

In spite of what we just said, sometimes you find you *have* use higher-level factors just to group a mixed bag, like this:

Politics; increase in populism

Politics; shift to the left

Politics; shift to the right

Politics; more engagement from younger voters.

The higher-level factor Politics is not in any sense a generalisation of these very disparate factors. However, we can at least think of it as a ‘mixed bag’. If we roll the map containing these stories up to level 1, we'll see this ‘mixed bag’ factor Politics as a cause and effect of other factors. It will be a bit hard to interpret, but we can live with it as long as we remember that it is a mixed bag rather than a semi-quantitative summary.

Hierarchical coding as a way of coping with a large number of factors

Usually analysts are faced with the quandary of either having too many factors which they lose track of, or merging them into a smaller number of factors and losing information. With hierarchies, you can have your cake and eat it; it is similar to the process of recoding an unwieldy number of factors into a smaller number of less granular items, but with the advantage that the process is reversible; the information can be viewed from the new higher level but also viewed from the original, more granular level. For example, we can count that the higher-level factor

component “Health behaviour” was mentioned ten times, while retaining the information about the individual mentions of its more granular components.

Don’t use a hierarchy when a hashtag will do

When the analyst wants to group certain factors into a theme (like “health”) which is not itself a causal factor, hierarchical coding should not be used. Instead, text hashtags like “#Environment” or “[Environment]” or just “Environment” can be used to create themes simply by adding the text hashtag to the factor label, e.g.

Soil loss (Environment)

Eco training courses for NGOs (Environment)

Re-usable factor components as hashtags

Sometimes your factors relate to each other in ways which are not just hierarchical. For example:

- Activities completed; Training; Health
- Activities completed; Distribution; Health; First-aid kits
- Outcomes achieved; Health; First-aid skills

These are three (hierarchical) factors in which “health” appears in different places, and at different levels of the hierarchy.

This is not ideal, but sometimes it’s just the best way to organise a tricky set of factors.

In this example, “Health” appears only as a “component” of other factors. Although on its own it might look like a mere theme rather than a causal factor, it plays a role in differentiating the causal factors in which it participates, e.g. “Activities completed, in particular training, in particular on health”; and because “Health” is used across hierarchies, it can *also* be treated as a [hashtag](#) and can be used as part of searches, lists and counts of factors, etc.

Isn’t that a contradiction? Didn’t we just say that “Health” is not to be used on its own as a factor because it is just a theme and is not expressed in a semi-quantitative way? No, because here the word “Health” does not function as only a theme but as a way of differentiating causal factors: all the actual factor labels in which it participates are correctly expressed in a semi-quantitative way. So Activities completed; Training; Health is intended as a causal factor about the extent of completion of activities, in particular training activities, and in particular health training activities.

Rigour and causal pathways

[!note]- Click to view the embedded note

Our approach is algorithmic

What we add: TODO

[Open page →](#)

[!note]- Click to view the embedded note Something else

Thanks guys, there's lots to like here and lots to agree with. Helping to find what causal hypotheses to focus on... What do you have to say about *pathways* as opposed to individual links/mechanisms? If we called this approach "quality and rigour in causal mechanisms evaluation" would that miss anything?

Disclaimer: for me, the logic around how links might combine into pathways and what that means for evaluation, that's the most exciting part. e.g. how might this intervention influence an outcome which might be multiple steps downstream of it?

Megan Shoji on cp coffee chat today 3/7/2025, getting ppl to rate the evidence for the whole pathway vs evidence for particular hotspots.

We've been honoured to be part of the Causal Pathways initiative since 2021, it's been really great to network with these evaluation experts, and we've learned and benefitted a lot.

As someone interested not only in the content of causal pathways work but also its logical underpinnings, I'm really glad to have Tom Aston and Marina Apgar's "How do we define and support quality and rigor in Causal Pathways evaluation?" (Apgar & Aston, 2025) as a go-to guide for this: Alongside (Lynn et al., 2021) I think it's the main Causal Pathways text.

BUT I think there is still something essential which is missing from Causal Pathways thinking as it currently stands.

Why do we talk about causal pathways (and causal networks and diagrams and maps) rather than just listing *individual links*: separate, hypothesised, cause/effect relationships?

What is it about the logic of causal *networks* which we are leveraging, or at least implying, when talking about causal pathways? Do we have guidance on how to assemble evidence for $A \rightarrow C$ on the basis of evidence for $A \rightarrow B$ and $B \rightarrow C$?

Assembling evidence to make evaluative judgements about *individual links*

One thing we do often do is query the network to answer questions like: what are the various influences on factor F.

```
graph LR
    Influence1 --> Some_factor
    Influence2 --> Some_factor
    Influence3 --> Some_factor
```

-or what are its various consequences.

```
graph LR
    Some_factor --> Consequence1
    Some_factor --> Consequence2
    Some_factor --> Consequence3
```

Evaluation literature

Occasionally we also mention network measures like centrality.

Looking at Marina and Tom's causal pathways "Questions" (for the CP training - below) - they never really talk about the fundamental property and challenge of a causal *map* which is **transitivity** or **tracing**.

From

```
a-->b
```

and

```
b --> c
```

concluding, (maybe, but when and how?),

```
a --> c.
```

They just never address the question. But if you weren't interested in this fundamental property & challenge, why would you use a map or talk about a pathway?? Causal questions that fit within this overarching question may focus on exploring causal pathways or on assessing impact. Apgar and Aston categorize them as follows and provide the following examples:

Aston and Apgar list "all possible types of causal questions." (Apgar & Aston, 2025, p. 16)

I've marked with ☒ those which I believe can be reasonably answered through a "single link" lens

Causal Process Questions:

- ☒ What specific processes led to the observed outcomes?
- ☒ How did the intervention trigger change?
- ☒ What intermediate mechanisms connect the intervention to outcomes?

Causal Context Questions:

- ☒ How do different contexts modify the causal mechanism?
- ☒ What contextual conditions enable or inhibit causal effects?
- ☒ How do local factors interact with intervention strategies?

Emergent Causality Questions:

- How do unintended consequences emerge?
- ? What unexpected causal pathways developed?
- ☒ How do multiple factors interact to produce outcomes?

Comparative Causal Questions:

- ? Which intervention strategy was most effective in creating change?
- ? How do different approaches compare in terms of causal impact?
- ? What are the differential effects of alternative interventions?

Equity Focused Questions:

- How did causal effects differ across social groups?
- What variations in impact existed?
- Did the intervention affect different populations differently?

Systemic Causality Questions:

- How did the intervention contribute to systemic changes?
- What broader transformations were triggered?
- How do multiple levels of the system interact causally?

Impact Contribution and Inquiry Questions:

- To what extent did the intervention cause the observed changes?
- What difference did the program make compared to doing nothing?
- How much of the observed outcome can be directly attributed to the intervention?

"The practice of critical reasoning in causal analysis involves: Identifying causal hypotheses and interrogating each step within a causal chain"

Transitivity, Janus

There are at least three problems of transitivity which we need to think about

- Given that A influences B and B influences C, does A influence C?

- Given that P believes that A influences B and P believes that B influences C, does P believe that A influence C?
- **Given that someone believes that A influences B and someone else P believes that B influences C, does someone (who? we? the people?) believe that A influence C?**

From eval24 However, we argue that evaluators can break the Janus dilemma and make the best use of causal maps in evaluation by considering causal maps not primarily as models of either beliefs or facts but as repositories of causal evidence. We can use more-or-less explicit rules of deduction, not to make inferences about beliefs, nor directly about the world, but to organise evidence: to ask and answer questions such as:

- Is there any evidence that X influences Z?
- . . . directly, or indirectly?
- . . . if so, how much?
- Is there more or less evidence for any path from X to Z compared to any path from W to Z?
- How many sources mentioned a path from X to Z?
- . . . of these, how many sources were reliable?

We also argue that this is a good way of understanding what evaluators are already doing: gathering and assembling data from different sources about causal connections in order to weigh up the evidence for pathways of particular interest, like the pathways from an intervention to an outcome.