Magnetisation

23 Dec 2025

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: Working Papers; Minimalist coding for causal mapping; Combining opposites, sentiment; 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.

1. Why magnetisation exists (the practical problem)

If you do open-ended coding, especially "radical zero-shot" AI coding, you get:

- high recall (lots of causal claims captured),
- but also a **hairball** of thousands of near-duplicate labels.

You then need some way to standardise labels so that maps and tables become readable *and* queryable.

There are two broad routes:

- **Auto clustering**: let an algorithm discover groups (often useful for exploration).
- **Magnetisation**: you decide the groups (magnets) and the algorithm assigns labels to them (best when you already know the vocabulary you want to use).

Magnetisation is basically "codebook application", but done **softly** (by semantic similarity) rather than by exact string matching.

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 vector encoding of meaning. 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). Semantic similarity is measured by cosine similarity (the angle between embedding vectors).

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 have high cosine similarity (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 $M_1, \ldots, M_k$ and a raw label L:

1. Compute similarity between L and each M_i.
2. Let M^* be the most similar magnet.
3. If similarity$(L, M^*) \geq$ 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:

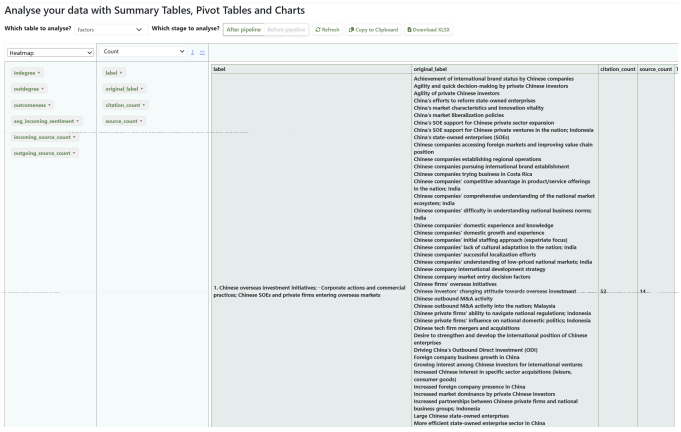
- you want a magnet to express a general theme (e.g., “health behaviours”),
- but the best magnet for matching may need to stay closer to the single-case language actually used in your corpus.

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.

Appendix: 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

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., … Amodei, 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.