Ex 2.2.7: a pilot of the syntax embeddings

> A scouting run into why the embeddings of the op words and `=` hold part of the anchor axis, which is what a full-position projection pays for on the non-red lines.
>
> The readout puts it there, to predict `=` after a red operand, and the tied table passes it to the embedding. Given a readout table of its own, the model keeps the component on that table and the syntax embeddings come mostly clean. Leaving the embeddings unanchored does not clean them.
>
> Untying costs nothing on task or placement, and brings the non-red cost of the projection down toward the cost of the operand-only edit. The pilot proposes it for the handover, since it needs nothing from the grammar. Under the whole-line labeller a few seeds lose selectivity, so that labeller should go in with a check rather than by default.

Observations

- **Where the component works.** On the stored `recipe-short` checkpoints, stripping it from the readout side costs the `=` prediction on the red lines (1.00 to 0.85; 0.16 on `t00`) and nothing else. Stripping it from the embedding side costs the answer instead (1.00 to 0.55 on the red lines, 0.82 on the non-red). But turning the embeddings by the same angle in a random direction costs at least as much (0.41), so on that side the blocks seem to have learned where the `=` embedding sits. The pilot arms repeat the pattern. On `untied`, stripping the readout side costs the `=` prediction (0.86) and stripping the embedding side costs little (0.95 on the answer). On `syntax-off-axis` and `untied-line`, no strip changes anything (figures).
- **The embedding-component table.** The `=` embedding is at 0.26 on `recipe-short` and 0.93 on `t00`; the op words are at 0.13 and 0.34. Anchoring the blocks only leaves the `=` embedding at 0.25. Untying the readout brings the `=` embedding down to 0.12, and puts 0.31 on the readout vector. Under the whole-line labeller the `⏎` embedding takes the axis as well (0.18 on `blocks-only-line`, 0.28 on the readout of `untied-line`). `syntax-off-axis` holds every syntax embedding at zero by construction. The red color embeddings themselves sit at 0.81 on the reference, 0.79 on `syntax-off-axis` and 0.94 on `untied`, but at 0.49 on `blocks-only` and 0.47 on `blocks-only-line`, where nothing pulls the embedding (figure).
- **Task cost.** Held-out exact match is at least 0.998 on every op of every arm (table).
- **Placement.** m_line runs 0.335–0.400 across the pilot arms, against 0.392 ±0.020 on `recipe-short`; `blocks-only` is the low end, and one seed of `blocks-only-line` did not place (m_line 0.12, against 0.35 for its next seed). $\bar{\alpha}$ at op1, the containment read, is 0.13 on `untied` and 0.23 on `untied-line` against 0.08 on the reference, and retention on `untied-line` is 0.93 ±0.06 (figure).
- **Suppression and selectivity.** Under the full-position projection the non-red `mix` deficit is 0.026 ±0.035 on `recipe-short`, where the operand-only edit gives 0.007. On the other arms it is 0.015 ±0.023 on `untied`, 0.011 ±0.024 on `syntax-off-axis` and 0.017 ±0.017 on `blocks-only`. Red-line `mix` accuracy under the projection is 0.01 on the reference, 0.02 on `untied` and 0.02 on `syntax-off-axis`. On `blocks-only` it is 0.17, with one seed at 0.90, and across the six ops the two blocks-only arms sit at 0.15–0.44 where the reference sits at 0.01–0.18. Under the whole-line labeller the non-red cost has a tail: three of nine `untied-line` seeds and one of nine `blocks-only-line` seeds lose more than 0.07 of the non-red `mix` lines under the projection, against one of the twenty reference seeds, and `untied-line` pays 0.019 ±0.032 under the operand-only edit as well (figure).

Why, and what we ran

Ex-2.2.2's E8 found the anchor axis on the embeddings of the op words and `=`, at a few tenths in every anchored model, and ex-2.2.3 found more of it at the heavier adopted point. That component is what a full-position projection pays for on the non-red lines, and the operand-only edit routes around it only because this grammar tells us where the operands are. In M3 there is no operand position, so the M3-shaped operator is full-position and needs the syntax embeddings clean. The tied-readout item and the design's Prep C each name a mechanism; this pilot runs both, with a third beside them.

nGPT ties the readout to the embedding, so one vector does two jobs: it is the residual stream's starting state when its token is read, and it is the logit of its token at every position that predicts it. Three mechanisms could put the axis on it.

1. **The direct pull.** The anchor term acts at every residual slice, the embedding included, and the labelled span covers the op word and `=`. Their slice-0 states are the embeddings themselves.
2. **The tied readout.** After a red operand the stream sits near e_1; the cheapest way to raise the logit of the token that follows is to lean that token's embedding the same way.
3. **The blocks.** A block that reads the component from the stream at a syntax position has a reason to keep it there, whichever table put it in.

Part A prices the component without training anything. On ex-2.2.3's stored checkpoints (`recipe-short`, 20 seeds, `t00`, 5 seeds, `control-short`, 5 seeds) we strip the axis component from the syntax embeddings on the *input* side only (the embedding, with the original table kept as the readout), the *output* side only, or both, and read the next-token accuracy at each position on the red and non-red probe lines, and the full-position projection's cost on the stripped model. A control turns the same embeddings by the same angle along a random direction off the axis, to read how much of an input-side cost is the turn itself. Whichever side loses the syntax predictions is where the component works.

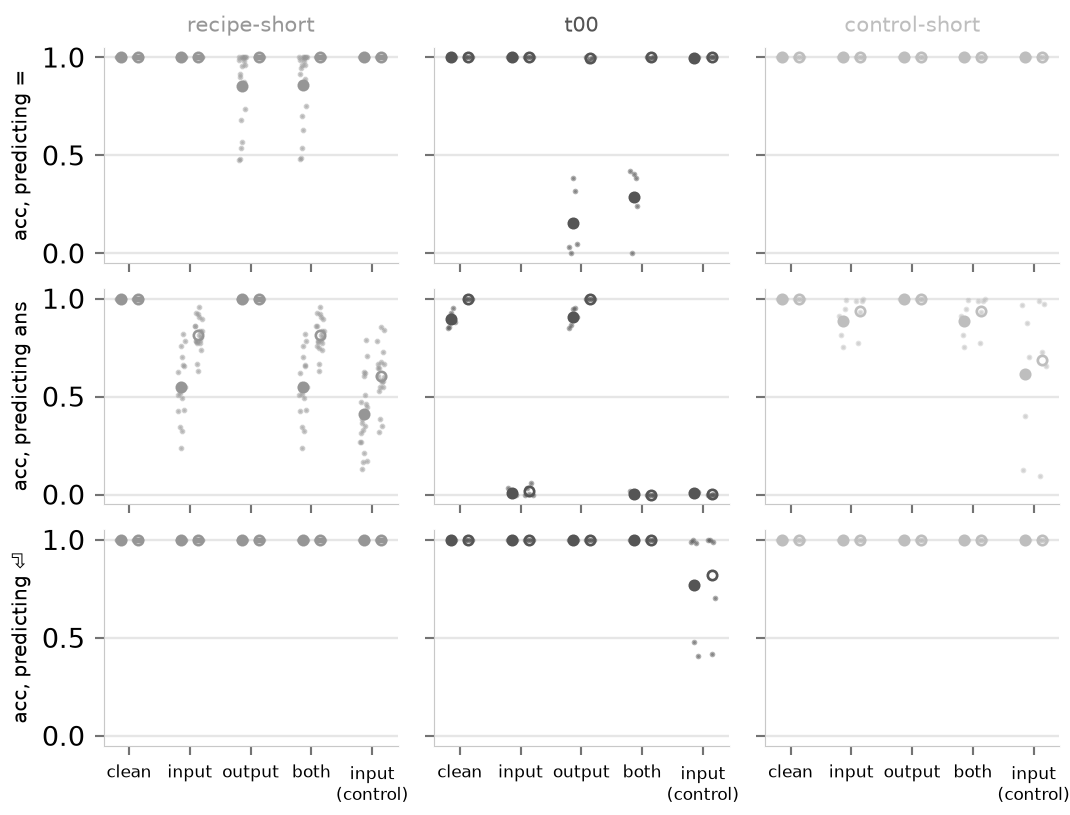
Part B retrains ex-2.2.3's `recipe-short` ($\lambda = 0.1$, 50 epochs) three ways, each removing one mechanism, and reads Part A's table, E8's embedding-component table, and ex-2.2.3's placement and suppression statistics on every run. The first two also run under ex-2.2.6's whole-line labeller, which the handover proposes to adopt and which pulls the syntax positions by design.

arm	seeds	readout	anchored slices	syntax held off axis	labeller	
`blocks-only`	9	tied	blocks only	no	operands	both terms skip the embedding slice (Prep C)
`untied`	9	untied	all	no	operands	a readout table of its own, from a copy of the embedding
`syntax-off-axis`	9	tied	all	yes	operands	tied table; the syntax embeddings held off the axis every step
`blocks-only-line`	9	tied	blocks only	no	whole line	blocks-only, under the whole-line labeller
`untied-line`	9	untied	all	no	whole line	untied, under the whole-line labeller
`recipe-short`	20	tied	all	no	operands	production, ex-2.2.3's recipe

`syntax-off-axis` (stored as `rows-clean`) is the simplest fix that keeps the table shared: after every optimizer step, the same projection that keeps nGPT's embeddings at unit length also zeroes the axis component of every non-color embedding. `untied` gives the readout a table of its own, initialised as a copy of the embedding, so the anchor and anti-subspace terms keep acting on the embedding and the logits are free. `blocks-only` is Prep C: both anchoring terms skip slice 0.

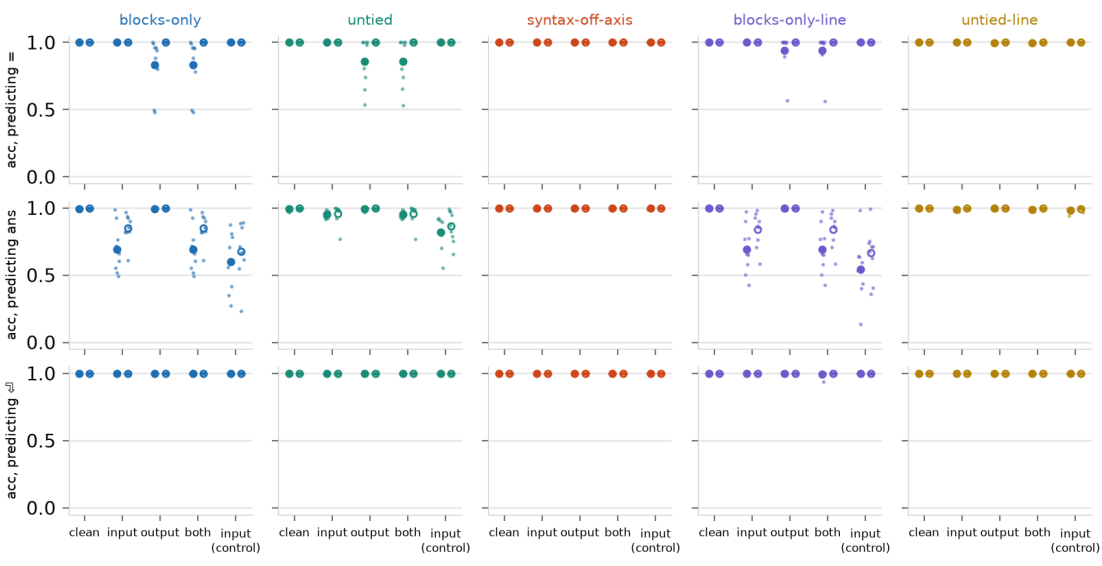
Where the component works

Part A on the stored checkpoints. Each panel is one of the three informative next-token predictions on the `mix` probe lines: `=` from op2, the answer from `=`, and the newline from the answer. The x axis is the strip condition. A filled marker is the red lines (both operands' redness at least 0.8), a hollow one the non-red lines; the small dots are the seeds and the larger marker their mean.



Next-token accuracy on the `mix` probe lines under each strip, on the stored checkpoints. *Rows are the predictions of `=`, the answer, and the newline; columns are the stored arms. Filled markers are the seed means on the red lines, hollow on the non-red, with one small dot per seed. `input` strips the axis component from the syntax embeddings and keeps the original table as the readout; `output` the reverse; `both` strips it from the shared table; the control moves the embeddings the same distance along a random direction off the axis.*

The same strips on the pilot arms. An arm whose syntax embeddings are already clean should show no difference across its strips; the untied arms split the two tables, so `input` and `output` strip different tables there.



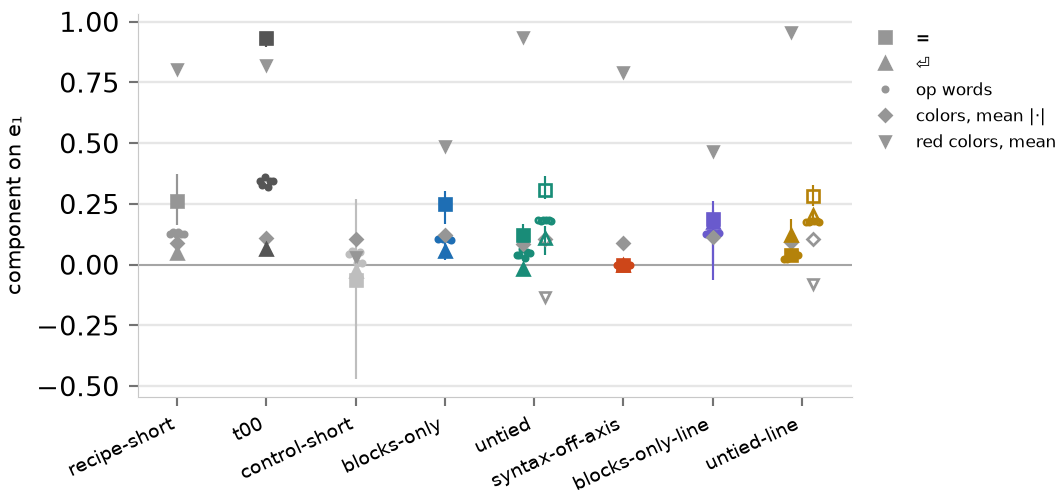
Next-token accuracy on the `mix` probe lines under each strip, on the pilot arms. *Same layout as the stored figure: rows are the predictions of `=`, the answer, and the newline; filled markers are the seed means on the red lines, hollow on the non-red, with one small dot per seed.*

arm	strip	answer acc, red	answer acc, non-red	P(answer), red	P(answer), non-red	projection: red acc	projection: non-red acc
`recipe-short`	clean	1.00 ±0.00	1.00 ±0.00	0.97 ±0.02	0.98 ±0.01	0.01 ±0.04	0.973 ±0.035
	input	0.55 ±0.29	0.82 ±0.17	0.52 ±0.26	0.76 ±0.16	0.01 ±0.04	0.973 ±0.035
	output	1.00 ±0.00	1.00 ±0.00	0.97 ±0.02	0.98 ±0.01	0.01 ±0.04	0.973 ±0.035
	both	0.55 ±0.29	0.81 ±0.17	0.51 ±0.26	0.75 ±0.16	0.01 ±0.04	0.973 ±0.035
	input (control)	0.41 ±0.33	0.61 ±0.27	0.37 ±0.30	0.53 ±0.30	0.03 ±0.09	0.716 ±0.352
`t00`	clean	0.89 ±0.05	1.00 ±0.00	0.74 ±0.04	0.97 ±0.00	0.00 ±0.00	0.396 ±0.365
	input	0.01 ±0.02	0.02 ±0.03	0.01 ±0.01	0.02 ±0.03	0.00 ±0.00	0.396 ±0.365
	output	0.90 ±0.05	1.00 ±0.00	0.76 ±0.04	0.97 ±0.00	0.00 ±0.00	0.199 ±0.245
	both	0.00 ±0.01	0.00 ±0.00	0.01 ±0.01	0.00 ±0.00	0.00 ±0.00	0.199 ±0.245
	input (control)	0.01 ±0.01	0.00 ±0.00	0.01 ±0.01	0.00 ±0.00	0.01 ±0.01	0.010 ±0.009
`control-short`	clean	1.00 ±0.00	1.00 ±0.00	0.97 ±0.01	0.98 ±0.00	0.97 ±0.04	0.955 ±0.052
	input	0.88 ±0.12	0.94 ±0.11	0.83 ±0.12	0.89 ±0.15	0.97 ±0.04	0.955 ±0.052
	output	1.00 ±0.00	1.00 ±0.00	0.97 ±0.01	0.98 ±0.00	0.97 ±0.04	0.955 ±0.052
	both	0.88 ±0.12	0.94 ±0.11	0.83 ±0.12	0.89 ±0.15	0.97 ±0.04	0.955 ±0.052
	input (control)	0.61 ±0.42	0.69 ±0.45	0.58 ±0.38	0.65 ±0.43	0.66 ±0.42	0.645 ±0.436
`blocks-only`	clean	1.00 ±0.01	1.00 ±0.01	0.96 ±0.03	0.98 ±0.02	0.17 ±0.45	0.980 ±0.018
	input	0.69 ±0.25	0.85 ±0.18	0.66 ±0.23	0.80 ±0.17	0.17 ±0.45	0.980 ±0.018
	output	1.00 ±0.01	1.00 ±0.01	0.96 ±0.03	0.98 ±0.02	0.17 ±0.45	0.980 ±0.018
	both	0.69 ±0.25	0.85 ±0.18	0.66 ±0.23	0.80 ±0.17	0.17 ±0.45	0.980 ±0.018
	input (control)	0.60 ±0.30	0.68 ±0.33	0.54 ±0.28	0.62 ±0.31	0.12 ±0.19	0.784 ±0.327
`untied`	clean	0.99 ±0.02	1.00 ±0.00	0.95 ±0.04	0.98 ±0.02	0.02 ±0.03	0.983 ±0.024
	input	0.95 ±0.03	0.96 ±0.12	0.89 ±0.04	0.92 ±0.13	0.02 ±0.03	0.983 ±0.024
	output	0.99 ±0.02	1.00 ±0.00	0.95 ±0.04	0.98 ±0.02	0.02 ±0.03	0.983 ±0.024
	both	0.95 ±0.03	0.96 ±0.12	0.89 ±0.04	0.92 ±0.13	0.02 ±0.03	0.983 ±0.024
	input (control)	0.82 ±0.18	0.86 ±0.17	0.77 ±0.18	0.82 ±0.18	0.02 ±0.03	0.909 ±0.114
`syntax-off-axis`	clean	1.00 ±0.01	1.00 ±0.00	0.97 ±0.01	0.98 ±0.02	0.02 ±0.02	0.987 ±0.022
	input	1.00 ±0.01	1.00 ±0.00	0.97 ±0.01	0.98 ±0.02	0.02 ±0.02	0.987 ±0.022
	output	1.00 ±0.01	1.00 ±0.00	0.97 ±0.01	0.98 ±0.02	0.02 ±0.02	0.987 ±0.022
	both	1.00 ±0.01	1.00 ±0.00	0.97 ±0.01	0.98 ±0.02	0.02 ±0.02	0.987 ±0.022
	input (control)	1.00 ±0.01	1.00 ±0.00	0.97 ±0.01	0.98 ±0.02	0.02 ±0.02	0.987 ±0.022
`blocks-only-line`	clean	1.00 ±0.01	1.00 ±0.00	0.97 ±0.04	0.98 ±0.01	0.15 ±0.19	0.927 ±0.282
	input	0.69 ±0.27	0.84 ±0.20	0.64 ±0.26	0.78 ±0.21	0.15 ±0.19	0.927 ±0.282
	output	1.00 ±0.01	1.00 ±0.00	0.97 ±0.04	0.98 ±0.01	0.15 ±0.19	0.927 ±0.282
	both	0.69 ±0.27	0.84 ±0.20	0.64 ±0.26	0.78 ±0.21	0.15 ±0.19	0.927 ±0.282
	input (control)	0.54 ±0.42	0.66 ±0.32	0.49 ±0.40	0.59 ±0.32	0.11 ±0.16	0.780 ±0.225
`untied-line`	clean	1.00 ±0.01	1.00 ±0.00	0.97 ±0.02	0.98 ±0.02	0.00 ±0.01	0.925 ±0.197
	input	0.99 ±0.01	1.00 ±0.00	0.95 ±0.02	0.98 ±0.02	0.00 ±0.01	0.925 ±0.197
	output	1.00 ±0.01	1.00 ±0.00	0.97 ±0.02	0.98 ±0.02	0.00 ±0.01	0.925 ±0.197
	both	0.99 ±0.01	1.00 ±0.00	0.95 ±0.02	0.98 ±0.02	0.00 ±0.01	0.925 ±0.197
	input (control)	0.99 ±0.03	0.99 ±0.02	0.94 ±0.04	0.97 ±0.02	0.00 ±0.01	0.904 ±0.198

The answer prediction on the `mix` probe lines under each strip, per arm, and the full-position projection read on the stripped model: exact-match accuracy on the red lines (the removal read) and on the non-red lines (the selectivity read, as an accuracy rather than a deficit, since a strip moves the clean baseline too). Seed means with half the seed range. The pilot arms follow the stored ones; a fix that has already cleaned its syntax embeddings shows no difference across its strips.

The embedding-component table

Ex-2.2.2's E8, read on every arm: the axis component of each syntax embedding, with the mean absolute component over the 216 color embeddings beside it, and the mean signed component of the red color embeddings (redness at least 0.8), which is how far the red operands themselves sit along the axis before any block runs. nGPT's embeddings are unit vectors, so a component is a cosine. The untied arms carry a second table; its readout vectors are drawn hollow.



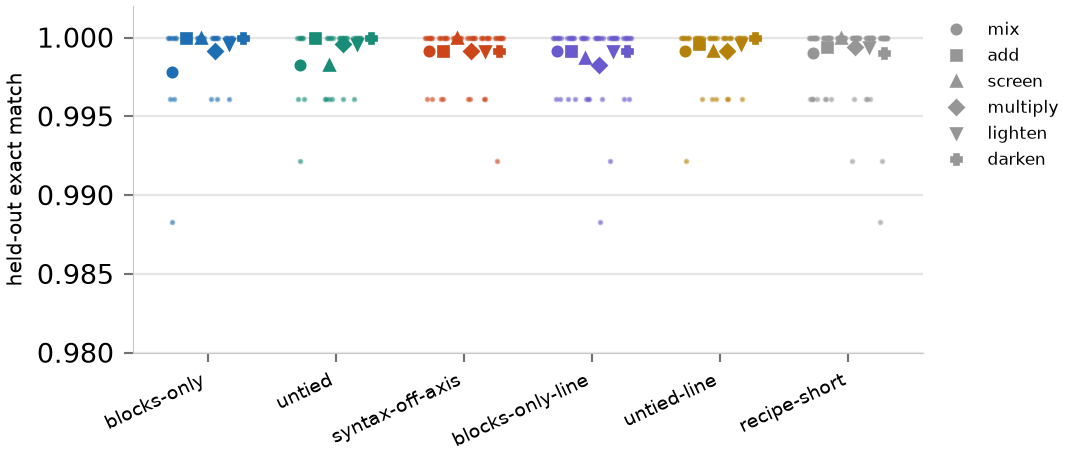
Axis component per syntax embedding, by arm. *Each marker is the seed mean of one embedding's component on e_1 (a cosine, since the embeddings are unit vectors), with the seed range as a bar; the op words are drawn small, and in grey are the color embeddings' mean absolute component (diamond) and the red color embeddings' mean component (triangle). Hollow markers are the readout table of the untied arms. The stored arms are ex-2.2.3's; every pilot arm is Part B's.*

arm	table	=	⏎	`mix`	`add`	`screen`	`multiply`	`lighten`	`darken`	colors, mean |·|	red colors, mean
`recipe-short`	embedding	0.26 ±0.11	0.05 ±0.02	0.12 ±0.06	0.14 ±0.05	0.13 ±0.06	0.14 ±0.05	0.13 ±0.06	0.13 ±0.06	0.090 ±0.008	0.81 ±0.03
`t00`	embedding	0.93 ±0.03	0.06 ±0.03	0.34 ±0.06	0.33 ±0.06	0.36 ±0.07	0.32 ±0.02	0.34 ±0.06	0.34 ±0.06	0.111 ±0.020	0.82 ±0.04
`control-short`	embedding	-0.06 ±0.37	-0.03 ±0.07	0.04 ±0.19	0.05 ±0.19	0.05 ±0.18	0.01 ±0.16	0.05 ±0.19	0.01 ±0.18	0.105 ±0.012	0.03 ±0.21
`blocks-only`	embedding	0.25 ±0.07	0.06 ±0.04	0.10 ±0.09	0.11 ±0.07	0.11 ±0.08	0.11 ±0.09	0.11 ±0.08	0.10 ±0.08	0.122 ±0.035	0.49 ±0.02
`untied`	embedding	0.12 ±0.04	-0.02 ±0.01	0.04 ±0.03	0.04 ±0.04	0.07 ±0.04	0.03 ±0.03	0.05 ±0.02	0.05 ±0.04	0.087 ±0.005	0.94 ±0.02
	readout	0.31 ±0.05	0.11 ±0.06	0.18 ±0.07	0.18 ±0.07	0.18 ±0.07	0.18 ±0.08	0.18 ±0.08	0.18 ±0.07	0.106 ±0.024	-0.13 ±0.11
`syntax-off-axis`	embedding	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.090 ±0.006	0.79 ±0.02
`blocks-only-line`	embedding	0.19 ±0.13	0.18 ±0.15	0.13 ±0.07	0.13 ±0.08	0.13 ±0.08	0.13 ±0.07	0.13 ±0.07	0.13 ±0.07	0.113 ±0.059	0.47 ±0.26
`untied-line`	embedding	0.04 ±0.03	0.12 ±0.05	0.02 ±0.04	0.02 ±0.02	0.04 ±0.03	0.03 ±0.04	0.05 ±0.04	0.04 ±0.04	0.094 ±0.004	0.96 ±0.01
	readout	0.28 ±0.04	0.21 ±0.03	0.18 ±0.06	0.18 ±0.06	0.18 ±0.06	0.18 ±0.06	0.18 ±0.06	0.17 ±0.06	0.105 ±0.039	-0.08 ±0.08

The embedding-component table as numbers: the signed axis component of each syntax embedding, per arm and table, seed mean with half the seed range; the last two columns are the mean absolute component over the color embeddings and the mean signed component over the red color embeddings.

Task cost

Exact match on the held-out pairs of each op. The reference is production's twenty seeds; the pilot arms have nine each.



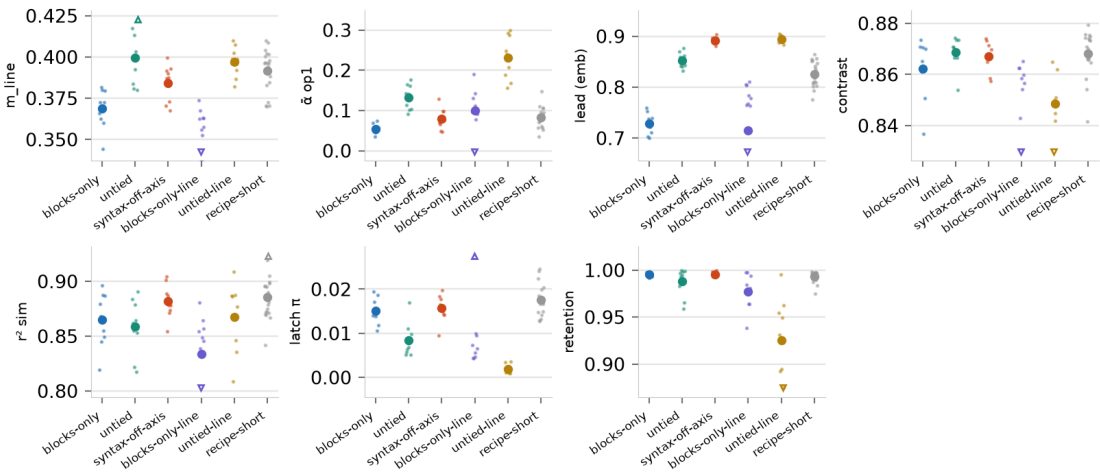
Held-out exact match per op, by arm. *One marker shape per op, the seed mean, with one small dot per seed; the y axis starts at 0.98. The reference is ex-2.2.3's production recipe.*

arm	seeds	`mix`	`add`	`screen`	`multiply`	`lighten`	`darken`	Δ mix
`blocks-only`	9	0.998 ±0.006	1.000 ±0.000	1.000 ±0.000	0.999 ±0.002	1.000 ±0.002	1.000 ±0.000	-0.001
`untied`	9	0.998 ±0.004	1.000 ±0.000	0.998 ±0.002	1.000 ±0.002	1.000 ±0.002	1.000 ±0.000	-0.001
`syntax-off-axis`	9	0.999 ±0.002	0.999 ±0.002	1.000 ±0.000	0.999 ±0.002	0.999 ±0.002	0.999 ±0.004	+0.000
`blocks-only-line`	9	0.999 ±0.002	0.999 ±0.002	0.999 ±0.002	0.998 ±0.006	0.999 ±0.004	0.999 ±0.002	+0.000
`untied-line`	9	0.999 ±0.004	1.000 ±0.002	0.999 ±0.002	0.999 ±0.002	1.000 ±0.002	1.000 ±0.000	+0.000
`recipe-short`	20	0.999 ±0.002	0.999 ±0.002	1.000 ±0.000	0.999 ±0.004	0.999 ±0.002	0.999 ±0.006	+0.000

Held-out exact match per op, seed mean with half the seed range; the last column is the `mix` gap from the production reference. Ex-2.2.3's task gate was a `mix` gap within 0.02 of its own control.

Where the pull lands

Ex-2.2.3's placement statistics on the `mix` probe lines, per arm. The line arms' m_line is under their own labeller's weighting, as in ex-2.2.6.



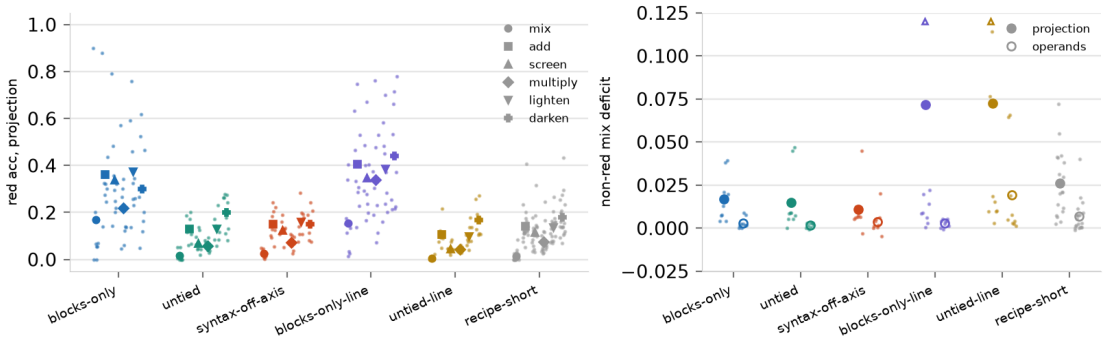
Placement on the `mix` probe lines, by arm. *One panel per statistic of the table below; the larger marker is the seed mean and the small dots are the seeds. Each panel spans the bulk of the seeds, and a seed beyond it is drawn as a hollow triangle at the edge (one `blocks-only-line` seed did not place, and sits off most panels; the table has it). The reference is ex-2.2.3's production recipe.*

arm	m_line	ᾱ op1	lead (emb)	contrast	r² sim	latch π	retention
`blocks-only`	0.369 ±0.019	0.054 ±0.019	0.73 ±0.03	0.86 ±0.02	0.865 ±0.038	0.02 ±0.00	1.00 ±0.00
`untied`	0.400 ±0.023	0.133 ±0.042	0.85 ±0.02	0.87 ±0.01	0.859 ±0.037	0.01 ±0.01	0.99 ±0.02
`syntax-off-axis`	0.384 ±0.016	0.079 ±0.040	0.89 ±0.01	0.87 ±0.01	0.882 ±0.025	0.02 ±0.01	1.00 ±0.00
`blocks-only-line`	0.335 ±0.129	0.100 ±0.114	0.71 ±0.33	0.76 ±0.43	0.834 ±0.099	0.12 ±0.49	0.98 ±0.03
`untied-line`	0.397 ±0.014	0.232 ±0.072	0.89 ±0.01	0.85 ±0.02	0.867 ±0.050	0.00 ±0.00	0.93 ±0.06
`recipe-short`	0.392 ±0.020	0.083 ±0.056	0.83 ±0.04	0.87 ±0.02	0.886 ±0.044	0.02 ±0.01	0.99 ±0.01

Placement on the `mix` probe lines, seed means with half the seed range. m_line is the per-line margin under the arm's own labeller; ᾱ op1 the mean alignment at op1 over every slice and color (ex-2.2.3's containment read); lead the G1 group's softmin weight on op1 at the embedding; contrast the deep-slice op2 weight of G2 minus G1; r² sim the grading of the op1 response against the similarity target; latch π the larger of the non-red group's deep-slice weights on the op word and on `=`; retention the final m_line over its running peak. On `blocks-only` the embedding slice is not pulled, so its lead is what the blocks' pull leaves there.

Suppression and selectivity

Ex-2.2.3's H4 statistics per arm: red-line accuracy under the full-position `projection` on each op (the removal read, lower is more complete), and the non-red `mix` deficit under `projection` and under the operand-only edit (the selectivity read). The question for each fix is whether the full-position deficit comes down to the operand-only one.



Suppression and selectivity, by arm. *Left, exact-match accuracy on the red lines under the full-position projection, one marker shape per op (the removal read; lower is more complete). Right, the non-red `mix` deficit under the projection (filled) and under the operand-only edit (hollow); the panel stops at 0.12, and a seed above it is a hollow triangle at the top edge (the table has the values). Larger markers are seed means, small dots the seeds.*

arm	`mix`	`add`	`screen`	`multiply`	`lighten`	`darken`	deficit, projection	deficit, operands	non-red acc, projection
`blocks-only`	0.17 ±0.45	0.36 ±0.37	0.34 ±0.32	0.22 ±0.26	0.37 ±0.31	0.30 ±0.28	0.017 ±0.017	0.003 ±0.004	0.980 ±0.018
`untied`	0.02 ±0.03	0.13 ±0.07	0.07 ±0.06	0.06 ±0.03	0.13 ±0.09	0.20 ±0.10	0.015 ±0.023	0.002 ±0.002	0.983 ±0.024
`syntax-off-axis`	0.02 ±0.02	0.15 ±0.09	0.13 ±0.07	0.07 ±0.04	0.16 ±0.10	0.15 ±0.08	0.011 ±0.024	0.004 ±0.012	0.987 ±0.022
`blocks-only-line`	0.15 ±0.19	0.41 ±0.31	0.35 ±0.25	0.34 ±0.34	0.39 ±0.27	0.44 ±0.28	0.072 ±0.282	0.003 ±0.003	0.927 ±0.282
`untied-line`	0.00 ±0.01	0.11 ±0.08	0.05 ±0.03	0.04 ±0.01	0.10 ±0.07	0.17 ±0.08	0.072 ±0.196	0.019 ±0.032	0.925 ±0.197
`recipe-short`	0.01 ±0.04	0.14 ±0.18	0.12 ±0.14	0.08 ±0.11	0.14 ±0.10	0.18 ±0.18	0.026 ±0.035	0.007 ±0.021	0.973 ±0.035

Exact-match accuracy under the full-position `projection` on the red lines (dose ≥ 0.8) of each op, then on `mix`: the non-red (dose ≤ 0.2) accuracy deficit under the full-position projection and under the operand-only edit, and the non-red accuracy under the projection. Ex-2.2.3's gates were red accuracy at most 0.2 on every op and a deficit at most 0.05.

What we make of it

Which mechanism. Each of the three arms removed one candidate, and two of the three results point the same way. Leaving the embedding out of the anchor did not clean the syntax embeddings, so the direct pull at slice 0 was not the cause. Giving the readout a table of its own moved the component onto that table, at about the size it has on the reference, and left the syntax embeddings mostly clean.

So the logit path is what places it. Part A says the same from the stored checkpoints: after a red op2, the `=` prediction leans on the component on the readout side, and that is the only thing the readout side does. The untied arm shows it from the other side: stripping the readout table costs the same prediction, and stripping the embedding costs almost nothing. This is the mechanism as written in the tied-readout item.

The embedding-side strips show something else: the blocks learn to read the embeddings where the logit path leaves them. The random-direction control costs as much, and on `t00`, where the `=` embedding sits about 70° off a clean one, a stripped model is a different model. So the component is doing work, and a stronger anti-subspace term on those embeddings would pull against the logits.

Prep C. Of the three hypotheses in the design, (a) holds in the blocks: the contrast and the similarity grading are at the values of the reference, with m_line a little lower. (b) holds, since the color embeddings still lead the pull at slice 0 even though nothing pulls them there. (c) does not hold, because the syntax embeddings keep the axis.

The cost Prep C did not anticipate is completeness. On both blocks-only arms the full-position projection leaves more red lines with their answer, on every op, and on one seed it leaves most of them. The embedding-component table says why: the projection removes the e_1 component of the stream at every slice and nothing else, so anything that survives it sits off the axis.

In the reference, the anchor at slice 0 and the anti-subspace term together put the redness of a red operand onto the axis before any block runs, with the red color embeddings at about 0.8 on e_1. With nothing pulling the embedding they sit at about half that, so half of the redness enters the stream off the axis.

The blocks are as well aligned as the reference's, since the contrast and the similarity grading match. They read the redness from the stream and write it onto the axis, and the projection removes what they wrote. The off-axis half is still in the residual stream at the last slice, where the readout can use it. So "the states at the later slices are aligned" describes what the blocks add, and how well that part is aligned does not decide whether the projection is complete.

What transfers. `syntax-off-axis` chooses its embeddings by token class: the non-color embeddings are the ones held off the axis. That is a rule about which vocabulary entries may hold the axis, the same kind of built-in position knowledge that the mellowmax pooling was adopted to avoid, and natural language has no such class. So the constraint as run here is a fix for this grammar, and what it measures is the ceiling: what a fully clean shared table buys.

The other two arms need nothing from the grammar. Untying is available on any model, as a copy of the table, and many models already ship untied. `blocks-only` keeps the table shared but pays in completeness, above.

A tied-table variant we did not run would hold every embedding off the axis, so the embedding has no axis component at all. It would need no token class, but the redness would then enter off the axis by construction, the completeness cost again. That question belongs with the operator pass, where the reflection and the shaped forms are also up for decision.

The fix to carry. On nine seeds, `untied` and `syntax-off-axis` match each other and the reference on task, placement and removal. Both have a lower non-red cost under the projection on most seeds, with a two-seed tail at the level of the upper seeds of the reference. The pilot proposes the untied readout for the handover, with the full-position projection read beside the operand-only edit, and `syntax-off-axis` as the in-grammar ceiling it should match.

The concern from the discussion in ex-2.2.3 still stands, that a method needing an untied readout is a harder sell where the tables are tied; the all-embeddings-off-axis variant above is the tied option to test if that becomes the target. Two things to watch on the untied arms: $\bar{\alpha}$ at op1, the containment read, is higher than on the reference, and the `=` embedding is lower rather than at zero.

The whole-line labeller. Under it the `⏎` embedding takes the axis the same way, since it follows the newly pulled answer, and on the untied arm it lands on the readout table like the rest.

The larger finding is the tail. On both line arms a few seeds lose a large share of the non-red lines under the projection, `untied-line` pays under the operand-only edit as well, and the margin on `untied-line` drifts down over training (the retention read). One `blocks-only-line` seed did not place at all.

There is no plain-recipe line arm at nine seeds here, and ex-2.2.6 has two or three, so the pilot cannot say whether the tail comes from the labeller alone or from pairing it with a fix. The handover should read the selectivity of the line labeller at more seeds before adopting it.

What it changes. The handover can adopt the untied readout, and use the full-position projection alongside the operand-only edit as its removal operator, which is what the M3-shaped operator needs. The operator pass in the D2.2 design chooses between plain projection and the shaped forms on that footing, and the whole-line labeller goes in with a selectivity check rather than by default.

Method notes

- Part A's strips edit the tables of a loaded checkpoint and nothing else; `input` and `output` give the tied model a readout table of its own for the scoring pass (`NGPT.with_tables`), so the two sides can differ. The stripped embeddings are re-normalized to unit length, as nGPT keeps them.
- The control strip replaces the axis component of each syntax embedding with one of the same size along a random unit direction orthogonal to e_1 (one fixed draw). The embedding then turns by about the same angle as under `input` and ends up off the axis as well.
- Part B's arms share ex-2.2.3's corpus, eval sets and probe lines (the same seeds through its `prepare_corpus`); the line arms train against ex-2.2.6's line-keyed probe table.
- `syntax-off-axis` (stored as `rows-clean`) applies `sca.anchoring.clean_embedding_rows` after every optimizer step, to every non-color embedding but the pad; `blocks-only` passes `anchor_slices=(1, 2, 3, 4)` to `train_anchored`, so both the anchor and the anti-subspace term skip the embedding; `untied` sets `tie_embeddings=False` on the model config, and its checkpoints carry the second table.
- The eval and score tasks are ex-2.2.3's, unchanged; on the untied arms the eval contract reads logits through the readout table and `ablate_weights` projects both tables.
- This is a pilot: five arms with nine seeds each (the first run had three, or two on the line arms; the rest were added on request, and the earlier seeds are memoized), Part A on five seeds of each frozen stored arm and on all twenty of the reference, and production's twenty for the reference elsewhere. Nothing here is gated, and nothing in it should be quoted as a result.