

Precision Is Not the Objective: Tuning Neural Knowledge-Base Completion for Downstream Interventional Reasoning

Miki
RatrektLabs
miki@ratrektlabs.com

Abstract—Neural knowledge-base completion is normally tuned on its own metrics: precision, recall, or ranking quality over held-out edges. We show that in a pipeline where a completion model feeds an exact symbolic reasoner, these objectives are anti-correlated over part of their range, and that tuning the completion model on them degrades the complete system. Raising the proposal-confidence threshold in our system nearly doubles proposal precision, from 31.8% to 58.3%, while downstream accuracy on interventional queries falls from 73.0% to 64.0%—barely above the 60.7% obtained by using no completion model whatsoever. The mechanism is asymmetric cost: a missing rule forfeits one answer, whereas aggressive filtering severs the causal paths the reasoner needs to reach any answer at all. We reach these conclusions on a deliberately small and fully reproducible system: a transformer of 819,968 parameters, with a 40-token vocabulary containing no natural language, trained from scratch on CPU in under twenty minutes, paired with a Datalog solver that implements intervention as literal graph surgery. Evaluating on 15,404 interventional queries over 200 held-out structural causal models, we additionally report that the neural component is worth 14.0 accuracy points over the solver alone—23.0 points on queries whose causal path was severed by knowledge ablation—and that under fixed knowledge the interventional operator outperforms conditioning by +7.9 and +7.6 points at two independent knowledge levels. A pretrained model 327 times larger, prompted zero-shot, scores 15.3% on the same queries, below both the majority-class baseline and uniform guessing. We state explicitly which of our numbers are structural rather than empirical, and release all code, seeds, and metrics.

Index Terms—causal inference, neuro-symbolic systems, do-calculus, knowledge base completion, logic programming, transformers

I. SCOPE AND CONTRIBUTION

We claim no novelty of mechanism. Neuro-symbolic architectures pairing a learned proposer with a symbolic reasoner are established [1], [2]; transformer-based knowledge base completion is a mature literature [3], [4], [5]; and the interventional calculus we implement is Pearl’s [6].

Our contribution is empirical. We evaluate a completion model not by link-prediction accuracy but by the accuracy of the interventional queries it enables, and find that the two objectives are anti-correlated over part of their range. Tuning the proposer to maximise proposal precision selects an operating point at which the complete system performs worse than with no proposer at all. To our knowledge this trade-off has not been reported for completion-plus-inference pipelines,

and it implies that component-level metrics are unsafe guides for tuning such systems.

We report this on a system small enough to be understood in its entirety and retrained in twenty minutes on a laptop CPU, with the comparisons that did not favour our system reported alongside those that did, and with two contaminations in our own metrics identified explicitly.

II. INTRODUCTION

A language model stores facts and reasoning ability entangled in the same weights. Facts cannot be edited, conclusions cannot be audited, and correcting one thing risks breaking another. An alternative division assigns each capability to the mechanism suited to it: a neural network generalises and fills gaps, while a symbolic engine chains exactly and produces proofs.

This paper studies that division on causal reasoning, where the distinction between mechanisms is unusually sharp. Deduction over a complete knowledge base is a solved problem—a Datalog engine derives every consequence exactly, with a proof, and a neural approximation of it would be strictly worse. The neural component can only earn its place where the symbolic one fails, and for a solver there is exactly one such place: knowledge it does not have.

We therefore construct a setting where incompleteness is controlled and measurable. We generate random structural causal models, ablate 30% of their rules, and ask whether a transformer trained to propose missing rules recovers enough structure for an exact solver to answer interventional queries it could not otherwise reach.

The second question we ask is whether the interventional operator itself is worth anything. It is easy to demonstrate that $P(Y \mid \text{do}(X))$ and $P(Y \mid X)$ differ on a hand-chosen example; it is more informative to measure the difference in aggregate, holding knowledge fixed, over models nobody selected.

A. Contributions

- A negative result on proposer tuning: optimising proposal precision in isolation selects a threshold at which the full system performs worse than with no proposer at all (Section V-E). This is the paper’s principal claim.

- A measurement of the neural proposer’s value, localised to the queries where the solver is structurally blind (Section V-A).
- A replicated estimate of the interventional operator’s value under fixed knowledge (Section V-B).
- A 2×2 evaluation design crossing knowledge completeness with reasoning operator, making each contribution attributable rather than confounded (Section IV).
- An explicit account of which reported numbers are definitional artefacts rather than empirical findings (Section VI).

III. METHOD

A. Worlds

We generate random structural causal models over $n = 10$ boolean variables $v_0, \dots, v_9$, of which the first $r = 2$ are exogenous roots sampled independently as Bernoulli(0.5). Each non-root variable v_i is assigned one or two rules; each rule body is a conjunction of one or two parents drawn from a locality window of the two immediately preceding variables. Multiple rules sharing a head denote disjunction.

Acyclicity holds by construction, since parents always have strictly lower index than their child. The locality window is not cosmetic: without it the induced graph is dense enough that almost every variable pair is one hop apart, and depth-4 queries occurred twice in 7,236 samples in an early configuration, leaving the depth analysis with nothing to measure.

With $r = 2$ there are only $2^r = 4$ root worlds, so all probabilities are computed by exhaustive enumeration rather than sampling. Every probability therefore lands exactly on $\{0, 0.25, 0.5, 0.75, 1\}$, which maps one-to-one onto our five answer buckets; bucketisation discards no information.

B. Solver and Intervention

Inference is naive forward chaining to fixpoint over ground boolean atoms, under the closed-world assumption that an atom absent from the derived set is false. The engine is roughly one hundred lines and returns both the derived set and the sequence of rules that fired.

Intervention is implemented as Pearl’s graph surgery, literally rather than approximately. To compute $\text{do}(X = v)$ we delete every rule whose head is X , then assert X if v is true:

```
def intervene(program, var, value):
    kept = [r for r in program
             if r.head != var]
    extra = {var} if value else set()
    return kept, frozenset(extra)
```

When the intervened variable is itself a root, surgery removes no rules, so the sampled root value must additionally be overridden. Writing $S(\cdot, \cdot)$ for the solver, $\mathcal{R}_{\setminus X}$ for the mutilated program, $w_v = (w \setminus \{X\}) \cup e_v$ for a root world with the intervention applied, and $\mathcal{W}_v$ for the root worlds in which X takes value v :

$$P(Y \mid \text{do}(X=v)) = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} \mathbb{I}[Y \in S(\mathcal{R}_{\setminus X}, w_v)] \quad (1)$$

$$P(Y \mid X=v) = \frac{1}{|\mathcal{W}_v|} \sum_{w \in \mathcal{W}_v} \mathbb{I}[Y \in S(\mathcal{R}, w)] \quad (2)$$

C. Representation

Knowledge bases and queries are serialised into a 40-token vocabulary: eight control tokens, four punctuation symbols, two truth values, ten variable names, five answer buckets, and eleven unused padding slots retained so the parameter count matches the figure used in our hardware budget. The vocabulary contains no natural language whatsoever.

Variables are canonically renamed per world, so the model must learn structure rather than vocabulary. Rule order is shuffled on every encoding: a knowledge base is a graph and admits no canonical linear order, and without shuffling the model learns to depend on the arbitrary ordering introduced by flattening.

Two example types share one format, with loss computed only on tokens following [ANS]:

```
completion
in : [BOS] v5 :- v4 . v2 :- v0 . ...
    [SEP] [MASK] [ANS]
out: v4 :- v3 . [EOS]

query
in : [BOS] v2 :- v0 . ... [SEP]
    [DO] v0 = 1 [QUERY] v3 [ANS]
out: B2 [EOS]
```

D. Model and Training

The model is a decoder-only transformer with 4 layers, $d_{\text{model}} = 128$, 4 heads, sequence length 128, and 819,968 parameters. It is trained from scratch; no pretrained weights are used.

Training from scratch is a deliberate methodological choice rather than a resource compromise. Pearl’s canonical examples—sprinkler and wet ground, smoking and tar—appear throughout the training corpora of publicly available language models. A pretrained model answering our queries correctly could not be distinguished from one recalling a memorised textbook example, which would render the confounding analysis uninterpretable. A from-scratch model evaluated on randomly generated graphs with canonical variable names has provably never encountered the test worlds.

We train on 2,000 worlds (22,317 examples) for 3,000 steps at batch size 64 with AdamW at learning rate 3×10^{-4} . Test loss falls from 3.62 to 0.44. Both objectives share one model, which keeps the neural-only ablation honest: it has identical capacity and identical training compute to the full system.

E. Completion with Confidence Filtering

At inference the model samples candidate rules, accepted only if they are syntactically well-formed, acyclic, novel, and sufficiently confident. Confidence is the mean per-token probability under the *untuned* softmax, so it reflects model belief rather than sampling temperature, and is length-normalised so short and long rules are comparable. Rules below a threshold $\tau = 0.5$ are discarded.

TABLE I
THE 2×2 DESIGN. READING ACROSS A ROW ISOLATES THE OPERATOR;
READING DOWN A COLUMN ISOLATES KNOWLEDGE.

Knowledge	do(X)	$P(Y X)$
Complete (oracle)	100% by constr.	D
Partial (70%)	A	E
Expanded (completed)	C (system)	F
Learned end-to-end	B	—
Pretrained, zero-shot	G	—

TABLE II
MAIN RESULTS. 15,404 QUERIES, 200 HELD-OUT WORLDS.
MAJORITY-CLASS BASELINE 42.1%.

	Knowledge / Operator	Acc.	MAE	Conf.
A	partial, do($\cdot$)	63.1%	0.223	22.2%
B	partial, learned	69.5%	0.191	19.0%
C	expanded, do($\cdot$)	76.1%	0.125	50.8%
D	complete, observe	79.2%	0.102	0.0%
E	partial, observe	55.2%	0.284	2.5%
F	expanded, observe	68.5%	0.174	2.2%
G	pretrained, zero-shot	15.3%	—	33.3%

Validation is what makes a fuzzy proposer safe to place in front of an exact solver: a malformed or cycle-inducing proposal is discarded rather than corrupting the knowledge base.

IV. EXPERIMENTAL DESIGN

Our evaluation crosses two axes: what a condition *knows* and what *operator* it applies (Table I).

The complete-knowledge, correct-operator cell is degenerate: it is the ground-truth generator, and therefore trivially perfect. It is reported as the ceiling that C approaches.

Conditions E and F are the crux of the design. Without them, the only correlational baseline is D, which reads the complete true model while A, B and C see only what survived ablation—so any C-versus-D comparison confounds knowledge with operator. E gives A a same-knowledge control, and F gives C one. **The C-versus-F gap is attributable to the operator alone.**

We report exact bucket accuracy and mean absolute error against the true interventional probability, using bucket mid-points so all conditions are scored on one scale. We stratify by reasoning depth (shortest directed path from intervened variable to target) and by whether a query is *confounded*, defined as $|P(Y | \text{do}(X)) - P(Y | X)| > 0.05$.

V. RESULTS

We evaluate on 15,404 queries over 200 held-out worlds. The majority-class baseline is 42.1%. In 38.5% of queries the partial knowledge base contains no directed path at all between the intervened variable and the target.

TABLE III
WHERE THE PROPOSER HELPS. MEASURED ON 60 HELD-OUT WORLDS.

Query subset	n	Solver (A)	+Proposer (C)	Gain
Path survives	2,824	72.7%	81.1%	+8.4
Path severed	1,772	44.1%	67.1%	+ 23.0
All	4,596	61.7%	75.7%	+14.0

TABLE IV
THE OPERATOR EFFECT, REPLICATED AT TWO KNOWLEDGE LEVELS.

Knowledge held fixed	Intervene	Observe	Gain
Partial KB	A 63.1%	E 55.2%	+7.9
Expanded KB	C 76.1%	F 68.5%	+7.6

A. The Value of the Neural Proposer

Table III localises the transformer’s contribution by splitting queries according to whether ablation severed the causal path.

The proposer’s value is concentrated precisely where the solver is blind. When a path survives, the solver can already chain through it and the proposer adds 8.4 points by supplying additional derivation routes. When the path is severed, the solver cannot reach the target at all and collapses to 44.1%; restoring the missing edge recovers 23.0 points.

This is the empirical answer to whether the neural component earns its place. It does—but only because the knowledge base is incomplete. Given a complete knowledge base the solver alone is exact, and the proposer would contribute nothing but error.

B. The Value of the Interventional Operator

Holding knowledge fixed and varying only the operator yields Table IV.

Two independent knowledge levels produce nearly identical gaps. Nothing in the construction forces this agreement, and we regard it as the strongest evidence in the study that the effect is attributable to the operator rather than to what each condition happened to know.

On confounded queries the separation is categorical rather than marginal. The full system scores 50.8%, while every correlational condition scores between 0.0% and 2.5% regardless of how much knowledge it holds—including D, which reads the complete true model. Correlation does not degrade gracefully on confounded queries; it fails outright.

C. Depth

Condition C is approximately flat across depth (−3.4 points from one hop to four), while the solver without completion collapses (−22.9 points) and the partial-knowledge correlational baseline collapses similarly (−9.8 points). Ablation severs long paths more often than short ones, and completion repairs enough of them to hold the full system level. Section VI qualifies the rising curves.

TABLE V
ACCURACY BY REASONING DEPTH (HOPS FROM INTERVENTION TO TARGET).

Condition	1 hop	2 hops	3 hops	4 hops
A: do($\cdot$), partial	72.4%	63.3%	56.1%	49.5%
B: transformer alone	68.1%	68.6%	70.2%	74.1%
C: do($\cdot$), exp.	78.0%	75.2%	74.9%	74.6%
D: observe, complete	74.5%	77.4%	82.1%	91.2%
E: observe, partial	58.2%	56.4%	52.9%	48.4%
F: observe, expanded	67.9%	68.1%	68.8%	70.4%

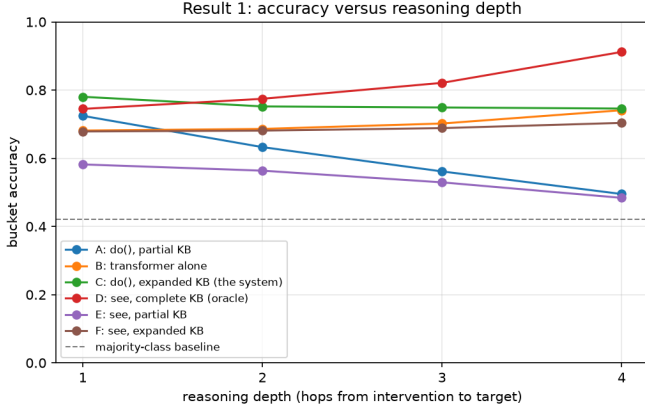


Fig. 1. Accuracy versus reasoning depth. The dashed line is the majority-class baseline.

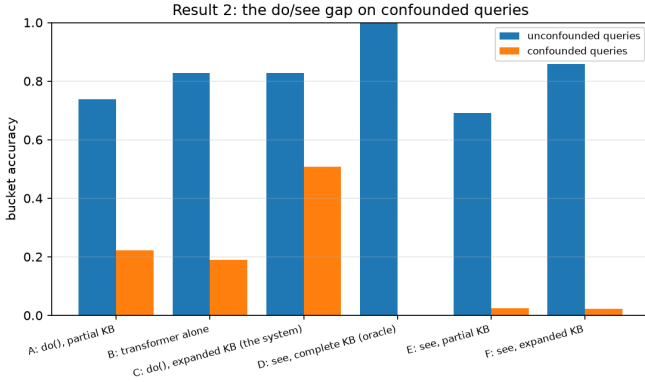


Fig. 2. Accuracy on confounded versus unconfounded queries.

D. A Pretrained Model of 327 Times the Size

Condition G asks whether a general pretrained model answers these queries without any of the machinery. Gemma 3 270M, instruction-tuned and 327 times larger than our network, was prompted zero-shot with the same knowledge base and query, and asked for the same five-way bucket.

It scored **15.3%** over 150 queries, against a 42.1% majority-class baseline and 20% for uniform guessing. Not one of its 150 answers was unparseable: it emitted a valid, confident label every time and was systematically wrong, which is a worse failure mode than refusing.

TABLE VI
THRESHOLD SWEEP ON TRAINING WORLDS. PRECISION AND RECALL OVER 280 PROPOSALS FROM 60 WORLDS; SYSTEM ACCURACY OVER 40 WORLDS. THE SOLVER WITHOUT ANY COMPLETION SCORES 60.7%.

τ	Kept	Precision	Recall	System acc.
0.00	280	31.8%	47.1%	73.0%
0.40	—	—	—	73.3%
0.50	202	38.1%	40.7%	77.2%
0.60	—	—	—	76.8%
0.70	83	51.8%	22.8%	74.0%
0.80	24	58.3%	7.4%	64.0%
0.85	5	20.0%	0.5%	—

Two caveats bound this comparison. It is zero-shot against a network trained on the task, so it measures what a general model does out of the box rather than its ceiling; a fine-tuned Gemma would be a different experiment. And the subset sizes differ, with the confounded subset here only 36 queries. What the number does establish is that scale alone, absent either the training or the solver, does not produce interventional reasoning at this task.

E. Proposer Tuning: A Negative Result

We selected the confidence threshold τ by sweeping downstream system accuracy on *training* worlds, never on the evaluation set. The sweep contradicts the objective one would naively optimise (Table VI).

Proposal precision rises monotonically with τ up to 0.80, where it reaches 58.3%—nearly double its unfiltered value. Yet downstream accuracy at that threshold is 64.0%, *below* the 73.0% obtained with no filtering at all, and only marginally above the 60.7% obtained using no proposer whatsoever.

The mechanism is straightforward in hindsight. A missing rule costs one answer; a spurious rule forges a path the solver will follow faithfully and corrupts every query routed through it. But filtering aggressively enough to maximise precision starves the solver of the edges it needs to connect any path at all, and an unreachable target is also wrong. The optimum balances the two, and only the downstream metric locates it. Optimising the proposer as a standalone component would have selected $\tau = 0.80$ and degraded the system.

Adopting $\tau = 0.5$ improved held-out system accuracy from 71.1% to 76.1%, accuracy on confounded queries from 40.9% to 50.8%, and MAE from 0.143 to 0.125.

VI. LIMITATIONS AND THREATS TO VALIDITY

We report two contaminations in our own metrics.

A. Condition D’s 0.0% is definitional

We define a query as confounded when $|P(Y | \text{do}(X)) - P(Y | X)| > 0.05$. On our probability grid, zero of 3,199 confounded queries place the two quantities in the same answer bucket. D returns $\text{bucket}(P(Y | X))$ by construction and is therefore guaranteed wrong on every such query. Conditions E and F exceed zero only because their imperfect knowledge occasionally lands in the correct bucket by accident. The

informative quantity is C’s 50.8%, which nothing in the design forces.

B. The depth axis measures two things

Answer entropy falls with depth, from 1.898 bits at one hop to 1.614 bits at four, as the rare buckets B1 and B3 shrink from 10.7% to 1.6% of queries. Deeper queries are therefore easier to guess. The majority-class baseline is flat across depth (42.2, 42.2, 42.1, 41.7%), so this is not simple class imbalance, but the rising curves for B, D and F should not be read as improved reasoning at depth. The falling curves for A and E, and the flatness of C, are not explained by this artefact and we regard them as sound.

C. The oracle wins overall

D achieves 79.2% against C’s 76.1%. Complete knowledge with the wrong operator still beats incomplete knowledge with the right one on average, because the 79.2% of queries that are unconfounded reward knowing more. We report this rather than omitting it; the appropriate comparison for C is F, where C leads by 7.6 points.

D. Scale and scope

Our worlds contain ten boolean variables and two exogenous roots. We do not claim these results extend to larger graphs, continuous variables, or noisy mechanisms. The answer distribution is skewed, with bucket B3 occupying under 1% of queries. All knowledge is synthetic, and the model handles no natural language; whether a pretrained encoder over verbalised knowledge preserves these effects is untested. This work addresses the second rung of the causal hierarchy, intervention. We do not implement counterfactuals, which additionally require abduction over exogenous variables conditioned on observed evidence.

VII. RELATED WORK

DeepProbLog [1] integrates neural predicates into probabilistic logic programming with end-to-end differentiability; Neural LP [2] learns first-order logical rules differentially for knowledge base reasoning. We deliberately avoid differentiable relaxation, training the proposer separately so each component can be evaluated in isolation—a choice that costs end-to-end optimisation but buys attributable ablations.

Knowledge base completion by embedding [3], by pretrained encoders [5], and by generative transformers [4] is mature, and our completion component is deliberately unremarkable within that line. What differs is the evaluation target. That literature reports ranking metrics over held-out edges—Hits@ k , mean reciprocal rank, precision—which score the completion model as an end in itself. We instead score it by the accuracy of the downstream interventional queries it enables, and Section V-E shows the two disagree over an important part of the operating range.

Our interventional semantics follow Pearl [6] directly. We address the second rung of the causal hierarchy only; counterfactuals additionally require abduction over exogenous variables conditioned on observed evidence.

VIII. CONCLUSION

Component-level metrics are unsafe guides for tuning a completion model that feeds an exact reasoner. Tuning our proposer for its own precision would have selected an operating point degrading the complete system to within three points of using no proposer at all, despite nearly doubling proposal precision. In architectures whose components are joined but not jointly optimised, only end-to-end measurement is trustworthy.

The remainder of our results establish that the pipeline is worth building in the first place. Separating proposal from inference is worth 14.0 points over the solver alone and 6.6 points over the neural model alone, with the gain concentrated where the symbolic component is structurally blind. The interventional operator is worth approximately 7.7 points under fixed knowledge, replicated at two knowledge levels, and is decisive rather than incremental on confounded queries.

REPRODUCIBILITY

All code, seeds, and metrics are available. The complete pipeline—data generation, training, evaluation, and figures—runs on an 8-vCPU CPU-only machine with no GPU. Training takes approximately nineteen minutes. The test suite comprises 75 tests covering oracle values from causal theory, order invariance, encoding round-trips, proposal rejection, and train/test separation.

REFERENCES

- [1] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt, “DeepProbLog: Neural probabilistic logic programming,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018.
- [2] F. Yang, Z. Yang, and W. W. Cohen, “Differentiable learning of logical rules for knowledge base reasoning,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017.
- [3] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 26, 2013.
- [4] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Çelikyilmaz, and Y. Choi, “COMET: Commonsense transformers for automatic knowledge graph construction,” in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2019, pp. 4762–4779, arXiv:1906.05317.
- [5] L. Yao, C. Mao, and Y. Luo, “KG-BERT: BERT for knowledge graph completion,” *arXiv preprint arXiv:1909.03193*, Sep. 2019.
- [6] J. Pearl, *Causality: Models, Reasoning, and Inference*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.