ESCROW: LEARNING GROUPS AND HIERARCHIES FROM STREAMING CATEGORICAL DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Categorical records arrive over time without supplied groups or parent relationships. A system must learn useful organization while avoiding groups that merely fit noise. ESCROW addresses this task through a data-source and industry-agnostic interface. Its central contribution is a complete conditional code that charges for groups, links, memberships, and cell owners. Each cell is encoded once, even when groups overlap. The code bounds unsupported selection under a stated no-association null; a separate result controls evidence accumulated from later blocks. Applied to COBWEB, selection raises mean membership F1 from 0.415 to 0.936 at depth one and from 0.622 to 1.000 at depth two, recovering all six reference groups and their links. It improves shallow recovery but lowers recovery at depths five through seven, and selects no groups on matched null streams. Recursive search separately achieves F1 above 0.999 in a 20-stream two-level confirmation. A semantic head shares sparse counts, and a memory head adapts trust in predictors. Matched experiments on Amazon and Wikidata compare classical, zero-shot, and few-shot LLM proposals. On Amazon, selected parent-based ESCROW saves 1.434 conditional bits per record, versus 1.032 for COBWEB and 0.705 for FISHDBC, with positive savings in all five orders. We measure structural recovery, conditional compression, and conventional prediction separately. Component studies isolate the two heads and ownership costs.

1 INTRODUCTION

Data rarely arrives as a complete, final collection. A system receives records, finds repeated patterns, and must revise its organization as more evidence appears. Those records may come from products, scientific observations, administrative systems, or knowledge bases. Their source can change while the learning problem remains the same: infer useful groups and relationships without supplied group labels, a fixed group count, or an industry-specific ontology. ESCROW is zero-shot with respect to supplied domain knowledge: it receives no labelled examples, domain instructions, or pretrained encoder. It learns counts and value similarities from the arriving records.

Consider records describing power source, connection type, and measurement unit. Repeated battery and wireless properties may support one group, with temperature sensors forming a subgroup. A rare combination may instead be noise. The system must distinguish useful structure from combinations that do not justify a group.

ESCROW separates proposing groups, selecting supported structure, and predicting later records. Repetition can shorten a description, but the groups, memberships, and links also cost bits. Our central contribution is a complete conditional code for overlapping categorical hierarchies, including explicit ownership of each record-field cell, with a bound on unsupported selection under a stated null. The semantic head shares evidence between values with similar contexts. The memory head adapts trust in the available predictors.

Recursive search and parent-based insertion offer two ways to propose groups. We compare their native and selected graphs with the same prediction heads and future records. COBWEB and FISHDBC anchor both recovery and real-data comparisons. Further classical methods and LLM prompts test the proposal interface on identical Wikidata inputs. Synthetic streams supply known hierarchies; real records test later prediction. The resulting comparisons show what search, selection, and prediction each contribute.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 FROM A USEFUL GROUP TO A COMPLETE DESCRIPTION

Records and groups. A record is a set of categorical field–value pairs. We treat an absent field as another symbol, written $\perp$, because absence can help distinguish groups. A graph D records group memberships, parent links, and the fields each group describes. Groups can overlap; a child can have several parents and must belong to each parent's set of records. Every record–field cell has one owner, either an eligible group or the background. This prevents overlapping groups from claiming the same saving twice.

Consider products with the fields `power`, `connection`, and `material`. A group may explain a repeated power–connection pattern while leaving material to the background. Another group may describe material for some of the same products. Membership alone does not entitle both groups to count the same field. The owner identifies which group pays for that cell.

What the comparison preserves. For a prefix x of n records, let T contain the count c_{av} of every symbol v in field a, including $\perp$. There are

$$N(T) = \prod_a \frac{n!}{\prod_v c_{av}!} \tag{1}$$

arrays with those column counts. Their set is the *type class*, $C(T)$. The reference length $L_0(x \mid T) = \log_2 N(T)$ describes only the arrangement. It does not reward a model merely for learning that one value is common.

What a group must pay. Our conditional description has four parts:

$$L_c(D, x \mid T) = L_{\text{graph}} + L_{\text{members}} + L_{\text{owners}} + L_{\text{cells}}(x \mid D, T). \tag{2}$$

The graph term includes nodes, links, and supported fields. Memberships are encoded jointly within families of groups with the same parents. The owner term records the route taken by each cell. Owned cells use normalized categorical codes; the remaining cells are an index into their remaining type class. The encoder and decoder follow the same order. Appendix B.2 specifies every term and the optional value code.

The support score is the saving

$$S(D, x) = L_0(x \mid T) - L_c(D, x \mid T). \tag{3}$$

We retain a selected graph only if $S \geq b$, where b is a declared evidence margin. At the default $b = 10$ bits, the one-prefix global-null error bound is $2^{-10} = 1/1{,}024$. This is neither a per-node error rate nor a learned group count. The empty graph has a one-bit structural charge and hence score -1.

A numerical example. Take two equally common record types, (x, u) and (y, v), and propose one group for each. With 40 records, the complete description costs 94.8 bits against a 74.0-bit reference, so the graph fails. With 80 records, the graph costs 141.2 bits and the reference costs 153.0 bits. The 11.8-bit saving now clears the margin. Repetition has paid for the groups. These are computed code lengths, not a search-recovery experiment.

Search proposes; the code evaluates. We evaluate two proposal mechanisms. Recursive search revisits groups and parent links at scheduled repairs, accepting complete-code improvements. Parent-based insertion updates a tree one record at a time, with child predictions smoothed toward their parents. They share the task but can produce very different candidate graphs. The conditional certifier can evaluate either pool. It selects fields and ownership, with optional pruning. Search can miss a useful graph even when the code would prefer it; the support rule does not guarantee that the best hierarchy was found.

This interface also accepts proposals from COBWEB, FISHDBC, or a language model. All charges remain in the final description, even if search uses a faster proxy. An ownership study holds overlapping memberships fixed and checks every legal route on small cases (Appendix F).

3 WHAT THE EVIDENCE GUARANTEES

The null hypothesis H_0 says that independently permuting each column leaves the law of the array unchanged. It contains iid records with independent fields. It is a statement about statistical association in this representation, not about whether human categories exist. Conditional on $T = t$, every array in $C(t)$ is then equally likely.

Theorem 1 (One prefix)**.** *Suppose the complete conditional description satisfies* $\sum_D \sum_{x \in C(t)} 2^{-L_c(D,x|t)} \leq 1$ *for every* t*. For every* $P \in H_0$ *and* $b \geq 0$*,*

$$\mathbb{P}_P\{\exists D : S(D, X) \geq b\} \leq 2^{-b}.$$

The bound concerns *any* graph that clears the threshold, so data-dependent search among encoded graphs is included. This is the familiar compression-testing argument applied to the complete graph description (Ryabko & Astola, 2006; Grünwald, 2007). Passing the test is evidence against this null, not proof that each group or link is semantically correct. Repeated tests of overlapping prefixes at a fixed threshold do not inherit this bound.

Evidence on fresh blocks. Divide the stream into blocks at predetermined boundaries. Before block k, choose a model using only the past. Conditional on the block's counts T_k, let Q_k be a normalized distribution on $C(T_k)$. It may adapt to cells already decoded within that block. Define

$$\Delta_k = \log_2 N(T_k) + \log_2 Q_k(X_k \mid T_k), \qquad W_k = \prod_{j=1}^{k} 2^{\Delta_j}, \quad W_0 = 1. \tag{4}$$

Theorem 2 (Fresh-block evidence)**.** *Assume that, given the past and* T_k*, block* X_k *is uniform on* $C(T_k)$*. Then* (W_k) *is a nonnegative supermartingale and* $\mathbb{P}(\sup_{k \geq 0} W_k \geq 2^b) \leq 2^{-b}$*.*

The proof is a conditional normalization check followed by Ville's inequality (Ville, 1939; Ramdas et al., 2023). It allows continuous monitoring at completed block boundaries. The process can be held constant between boundaries; it does not obtain new certified evidence from an unfinished block. This controls a global-null alarm, not the correctness of later individual node additions after real structure is already present.

A normalized block code. Let $r_a(v)$ be the number of copies of value v still unencoded in column a. A positive model weight $M_{ia}(v)$ gives the next-cell distribution

$$q_{ia}(v) = \frac{r_a(v) M_{ia}(v)}{\sum_u r_a(u) M_{ia}(u)}. \tag{5}$$

Each draw removes one copy. With $M = 1$ this is uniform over $C(T_k)$; a useful model changes which arrangements are probable. Mixtures of such predictions remain normalized. The block counts are available for this conditional assessment, so $\log_2 W_k$ is not the unconditional log-loss of a predictor that knows nothing about the new block.

Why support can be harder than prediction. In an ideal finite-alphabet model with fields independent given a group G, naming members has limiting saving $\sum_a I(X_a; G) - H(G)$ per record. A correctly specified mixture predictor instead gains $\sum_a I(X_a; G) - I(X; G)$. The difference is $H(G \mid X)$: uncertainty about which group a record belongs to. These are population rates under estimable component distributions, not a guarantee of finite-sample recovery. Appendix B proves both theorems and every supporting claim.

4 THE SYSTEM AND THE EVALUATION QUESTIONS

ESCROW accepts categorical field-value records without a source-specific schema or ontology. Its proposer returns a graph, the conditional code selects supported structure, and the semantic and memory heads predict later records. Recursive search and parent-based insertion are two proposer choices. External proposals use the same selection and prediction interfaces. Records arrive sequentially; scheduled repairs can revisit the stored prefix.

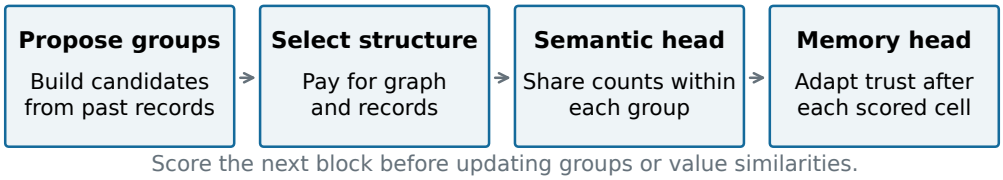


Figure 1: From arriving records to supported groups and predictions. Groups and value similarities use the past; predictor weights update after each scored cell.

Semantic head: share evidence without merging groups. A rare value borrows evidence from values that occur with similar other fields. Prefix co-occurrence counts define similarities that smooth counts inside each group. Groups keep their identity. No pretrained encoder or external data is used.

Memory head: learn which predictor to trust. For each field, the head weights group counts, semantic predictions, and the reference. After a cell, it rewards predictors that assigned it more probability. A .01 pull toward equal weights permits recovery from early losses. Weights reset at each evaluation block. Accumulated evidence W is a separate state that retains block gains and losses.

Table 1: Matched comparisons share inputs within each study. The appendix gives all stress tests, settings, repetitions, and implementation paths.

Study	Inputs	Purpose
Matched pipeline	42 depth cases; 40 real order/control cases	Both ESCROW searches, COBWEB, FISHDBC; recovery and both prediction measures
Wikidata breadth	2 variants, 5 orders, signal/null	Eight more classical proposers, zero/few-shot LLMs, and KNN
Recovery and noise	Flat noise ladder; 20 reserved streams per confirmation cell	Recovery, deferral, depth, and sample size
Components and support	Five head settings; 70 ownership cases; six margins	Heads, cell ownership, selection sensitivity, and cost

Synthetic nulls shuffle columns independently while preserving their counts. Real studies use Amazon Berkeley Objects (Collins et al., 2022) and Wikidata people (Vrandečić & Krötzsch, 2014), with and without hierarchy-revealing fields. These are four feature variants of two corpora. Five sampled orders vary records and ordering. Every graph comparison keeps native and selected outcomes separate. Conventional loss uses prefix-only vocabularies and records unseen values and excluded fields (Appendix D).

The ownership study supplies candidate memberships to isolate the code. The unchanged selector retains both groups on all 20 signal pools, spanning zero to 50% overlap, and selects none on 20 shuffled controls (Appendix F).

5 RECOVERING SIGNAL WHILE LEAVING NOISE UNGROUPED

Question and setup. A useful grouping rule should recover repeated structure and avoid inventing groups when that structure disappears. The classical noise ladder uses 3,000 records, eight latent groups, six categorical fields, and three seeds. For each field, a value comes from its group's slice of a shared inventory with probability s, otherwise from the whole inventory. At $s = 0$ no reference groups exist. We compare the flat ESCROW search with split moves against classical methods on the same records. Settings, oracle references, and label-free choices are distinguished in Appendix E.1.

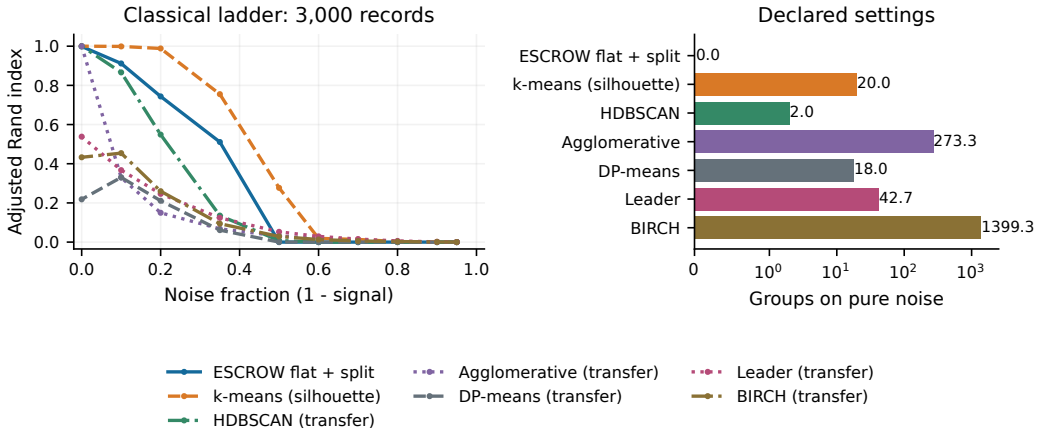


Figure 2: Classical noise ladder. Left: recovery as noise increases. Right: groups returned on pure noise. Three streams per condition. ESCROW combines strong full-signal recovery with no groups on pure noise. The complete tables retain every baseline setting and its recovery cost. ARI is omitted when no reference groups exist.

Recovery and restraint. At full signal, flat ESCROW with split moves reaches mean adjusted Rand index 0.9986; it returns zero groups on all three pure-noise streams. ARI treats unassigned records as one background cluster. Development-selected DP-means and BIRCH reach 0.2183 and 0.4329, respectively, at full signal. Silhouette-selected k-means reaches ARI 1.0 at full signal, but returns 20 groups on pure noise. This comparison shows the value of evaluating recovery and deferral together. A high recovery score alone does not establish when a system should create a group.

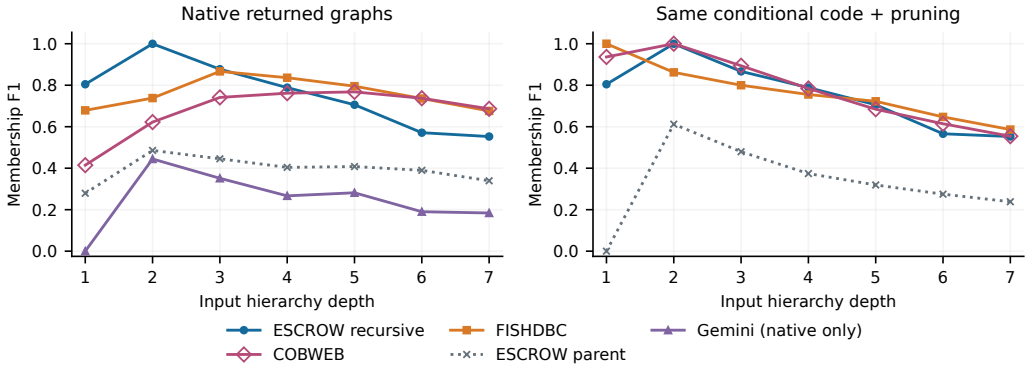
The tradeoff depends on signal strength. At $s = 0.65$, k-means has mean ARI 0.755, compared with ESCROW's 0.511. Some HDBSCAN settings also return no groups on noise; their signal recovery is reported alongside that behavior. The evidence supports a particular balance between recovery and restraint, rather than superiority at every noise level or a claim that all baselines invent structure.

Short streams with irrelevant fields. A separate stress test uses the parent-based proposer, a 300-record prefix, six informative fields, and six irrelevant fields. ESCROW, COBWEB, and FISHDBC all enter the same conditional certifier with pruning used in the depth comparison. All three return zero groups from signal 0 through 0.90. At full signal, their mean membership F1 values are 0.112, 0.545, and zero. This pipeline rejects noise but misses substantial signal. Its shorter input, added fields, search mechanism, and membership metric distinguish this stress test from the flat-search ladder.

A matched 300-record LLM ladder permits both recursive ESCROW and Gemini to defer; both return no groups at every rung. It supplies no recovery advantage for either arm. The complete short-stream curves, three-block predictive scores, and null controls remain in Appendix E.8; the LLM setup is in Appendix E.1. The matched real-data study also includes KNN as a prediction-only baseline.

6 RECOVERING GROUPS AND THEIR HIERARCHY

Question and setup. What do search and selection each contribute? Four proposers see the same 300-record streams: three seeds at depths one through seven, each with a column-shuffled null. Each level adds three informative fields; six fields are irrelevant. We compare each returned graph with the result of the same conditional code and pruning, using a 10-bit margin. Labels are reserved for evaluation.



Null groups: native / pruned	1	2	3	4	5	6	7
ESCROW recursive	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
COBWEB	119 / 0	126.3 / 0	118 / 0	108.3 / 0	105.7 / 0	100.3 / 0	103.7 / 0
FISHDBC	18 / 0	14.7 / 0	12.7 / 0	9.3 / 0	11.3 / 0	8 / 0	9.3 / 0
ESCROW parent	83 / 0	82 / 0	84.7 / 0	100.7 / 0	107.3 / 0	90 / 0	62 / 0
Gemini	0 / n/a	0 / n/a	0 / n/a	0 / n/a	0 / n/a	0 / n/a	0 / n/a

Figure 3: Matched recovery before and after conditional pruning. Colors identify the same proposer in both panels. "Native" retains each proposer's own search rules. Null cells show mean native / pruned group counts; n/a means not evaluated. Whole conditional selection, returned depth, and every case appear in Appendix E.3.1. The EHR05 replacement rule is reported separately.

Selection refines shallow external proposals. Conditional pruning raises COBWEB's depth-one to depth-three F1 from 0.415, 0.622, and 0.741 to 0.936, 1.000, and 0.894. At depth two, it selects exactly the six reference groups and all their links on every stream. Whole conditional selection rejects those same shallow pools. FISHDBC with pruning recovers depth one exactly, exceeding recursive ESCROW there. All four selected methods return no groups on the null streams. These cases show that large candidate pools can contain accurate, supported subsets; recursive search is not the best recovery configuration at every shallow depth.

Strong recovery from recursive search. Recursive ESCROW reaches native membership F1 0.805, 0.99972, and 0.877 at depths one through three. Whole conditional certification preserves all its signal graphs and all 21 empty nulls. A separate 20-stream depth-two confirmation gives F1 0.99945 and perfect parent and ancestor recovery. Recursive search exceeds selected parent-based insertion on all 21 matched signal streams. At depth one, selected parent-based insertion returns no groups on all three streams, giving F1 zero.

Deeper structure and the LLM probe. Pruning can improve shallow recovery while removing real groups deeper in a hierarchy. Native COBWEB and FISHDBC reach depth-seven F1 0.686 and 0.676, versus recursive ESCROW's 0.553. More data helps: its 20-stream depth-seven confirmation rises from 0.539 to 0.690 with 1,000 records. Gemini returns at most two levels and reaches F1 0.184 at depth seven. A compact-context probe reaches F1 0.753 on one shallow case. These are prompt-specific results; Appendix H gives the exact instructions.

7 PREDICTION ON REAL RECORDS AND LLM PROPOSALS

Question and setup. Both ESCROW searches, COBWEB, and FISHDBC use identical Amazon and Wikidata inputs: 600 training records and three 100-record future blocks in each of five sampled orders. Wikidata adds eight classical proposers, zero-shot and few-shot Gemini and Qwen, and prediction-only KNN. Graph methods share selection and prediction heads. Conventional loss uses prefix information and maps unseen values to an unknown symbol. Conditional compression instead knows the future block counts.

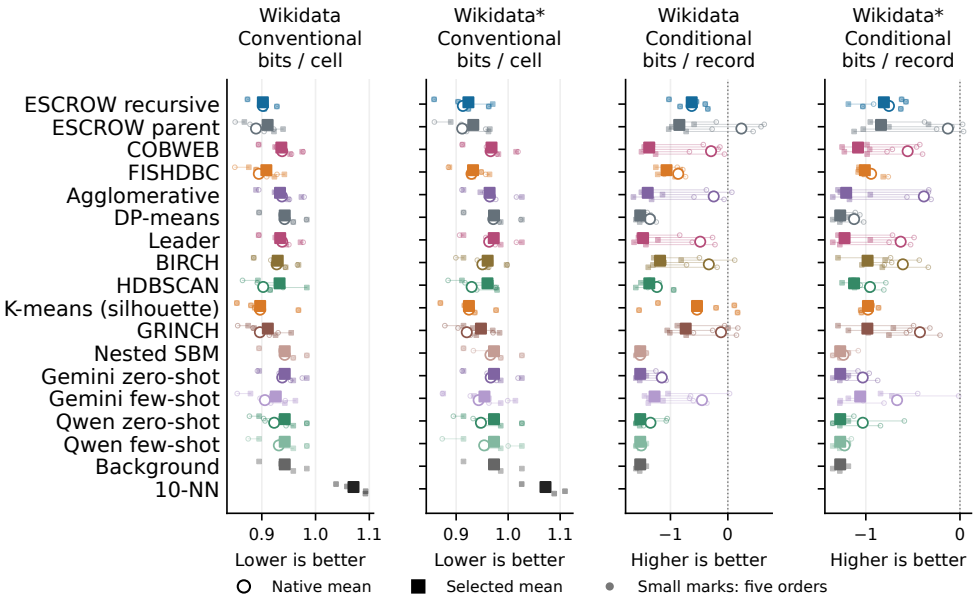


Figure 4: Wikidata on identical inputs. Left: conventional loss; right: conditional compression. Small marks show five orders, each averaging three blocks; lines pair native and selected outcomes. Large circles and squares show their means. * removes hierarchy-revealing fields. KNN supplies only conventional loss. Background uses the same heads; zero conditional saving denotes the untilted reference. Paired intervals: Appendix G.

What the records show. Selected parent-based ESCROW has the best mean Amazon scores among the tested selected graph methods. On both Amazon variants, both selected ESCROW searches beat COBWEB, FISHDBC, KNN, and background in conventional loss. Native parent-based ESCROW has the lowest mean conventional loss among native graph methods on both Wikidata variants.

For parent-based ESCROW, selection reduces mean conditional bits saved on all four variants: Amazon $1.688 \to 1.434$, Amazon* $0.950 \to 0.753$, Wikidata $+0.236 \to -0.845$, and Wikidata* $-0.129 \to -0.839$. Conventional loss moves differently: it improves from 2.112 to 2.056 on Amazon and from 1.930 to 1.879 on Amazon*, but worsens on both Wikidata variants. Every selected method selects no groups on matched Wikidata null streams. Selection tests structural support, not future predictive quality. Appendix D gives both paired changes.

What zero-shot and few-shot prompting contribute. ESCROW uses no domain instructions or demonstrations. LLMs are pretrained; their few-shot arm adds three worked bird examples unrelated to Wikidata. We report this demonstration-guided condition separately. Neither zero-shot model yields a selected signal graph. Few-shot Gemini retains groups on nine of ten signal inputs; Qwen retains none. Gemini improves mean prediction over empty groups but trails both selected ESCROW searches on both Wikidata variants. Qwen few-shot has 72.12% valid assignments and 166/600 JSON parse failures, so its outcome also reflects format compliance. These results apply to the recorded prompt and parsing protocol. The appendix gives exact prompts and every request diagnostic.

8 How recovered structure becomes useful prediction

One graph, two outcomes. Table 2 averages recovery and prediction on the same 21 signal streams, with shared semantic and memory heads. Selected COBWEB has the highest mean F1, 0.781, but improves on only 11 of 21 streams, with median gain 0.022. Large shallow gains drive its mean increase of 0.105; every depth-five to depth-seven stream loses recovery. FISHDBC and parent-based insertion each improve on 6 of 21 streams. Parent-based selection lowers mean recovery but improves mean prediction; FISHDBC shows the reverse. These averages describe different effects across depths and different outcomes for recovery, conditional saving, and conventional loss.

Table 2: Matched structure and prediction, averaging depths 1–7 and three seeds. F1 is membership recovery; bits/record is conditional saving; loss/cell is conventional log-loss. Higher is better for the first two, lower for the last. Selected means conditional pruning with $b = 10$. All four selected methods return zero groups on all 21 paired nulls.

Search	Graph	F1	Bits/record	Loss/cell
ESCROW recursive	Native	0.757	16.857	3.756
ESCROW recursive	Selected	0.755	16.826	3.758
ESCROW parent	Native	0.393	16.727	3.730
ESCROW parent	Selected	0.328	17.480	3.720
COBWEB	Native	0.676	16.434	3.744
COBWEB	Selected	0.781	17.520	3.724
FISHDBC	Native	0.761	16.516	3.764
FISHDBC	Selected	0.768	16.077	3.797

Separate contributions of the two heads. For each graph, five settings separate ordinary group prediction, semantic sharing, and fixed or adaptive expert weights. Adding the semantic expert improves mean conditional compression for both ESCROW searches on all four real-data variants. Adaptive memory lowers conventional loss in every search–variant mean. Its conditional effect differs: it helps on Wikidata, while the fixed mixture does better on Amazon. An independent scalar calculation confirms that tradeoff.

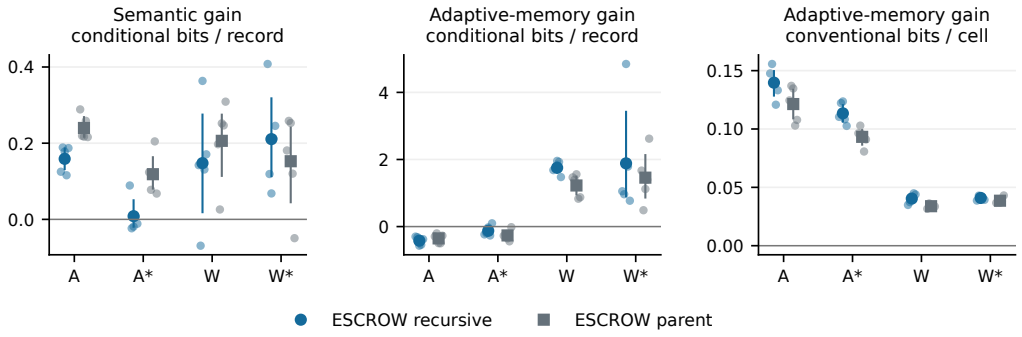


Figure 5: Controlled head changes on identical selected graphs. Positive values favor the added head. A: Amazon; W: Wikidata; *: removed fields. Large marks average five paired orders. Small points show each paired change; bars show descriptive 95% intervals. The middle and right panels compare adaptive and fixed weights over the same experts. Appendix G.2 shows how their weights change.

Selection margin and computation. Replaying $b = 2, 5, 10, 15, 20, 30$ leaves all 345 pruned decisions unchanged, including every null (Appendix D.10). In the repair ablation, retaining the candidate tree reduces mean runtime from 320.2 to 302.2 seconds; omitting repeated certification takes 24.0 seconds but provides no certificate. Matched recursive proposals take up to 4,923.8 seconds on 600 Wikidata records. These costs motivate incremental certification; they are not a controlled speed ranking across methods.

9 Related work

Proposing useful groups. COBWEB builds groups to improve prediction (Fisher, 1987). HDBSCAN, FISHDBC, PERCH, and GRINCH offer other candidate hierarchies (Campello et al., 2013; Dell'Amico, 2019; Kobren et al., 2017; Monath et al., 2019). DenStream keeps summaries of dense regions, discounts old observations, and separates potential clusters from outliers (Cao et al., 2006). It already addresses when streamed observations deserve a group. ESCROW adds a complete conditional description and a stated global-null guarantee for categorical record graphs.

Selecting model complexity. Minimum message length and minimum description length balance model size against fit (Wallace & Boulton, 1968; Grünwald, 2007). KRIMP and SLIM select itemsets by encoding both a code table and the transactions it covers (Vreeken et al., 2011; Smets & Vreeken, 2012). Their transaction covers account for each item once. ESCROW specifies a conditional code for hierarchical memberships and cell owners, with column counts held fixed, and derives its stated null bound from that complete code. X-means uses BIC (Pelleg & Moore, 2000; Schwarz, 1978); DP-means derives a creation penalty from a small-variance limit (Kulis & Jordan, 2012). Recursive Indian buffet process inference learns overlapping latent features (Schaeffer et al., 2022).

Turning compression into evidence. Coding tests supply the compression argument (Ryabko & Astola, 2006); Ville's inequality controls threshold crossing (Ville, 1939; Ramdas et al., 2023). Composite-null extensions use e-processes (Ruf et al., 2023). E-detectors combine evidence over possible change times and control average run length to a false alarm (Shin et al., 2024). That target differs from our probability of ever crossing a fixed global-null threshold. Restarting after evidence loss is a promising extension, but its guarantee needs a separate specification.

10 Scope and next steps

The matched pipeline connects search, selection, and prediction on identical inputs. Selection recovers shallow structure from external proposals; recursive search supplies strong shallow hierarchies. Selected parent-based insertion gives the best mean Amazon prediction among the tested selected graph methods. The ownership study establishes selection on supplied overlapping groups; hierarchy recovery separately tests discovered groups. Rankings depend on the corpus and selection rule. The certificate tests structural support; conventional loss measures future prediction.

Further work follows directly from these measurements. Deep hierarchies need better use of limited records: more data helps, while pruning can discard real lower-level groups. Persistent memory needs a study in which graphs change and weights carry across blocks; the present factorial comparison isolates adaptive weighting with resets. Incremental certification should reduce the cost of repairs that revisit the stored prefix. Additional corpora would test transfer beyond the two evaluated sources. All reported ESCROW configurations learn without supplied domain knowledge or demonstrations.

The theoretical targets remain precise. The conditional certificate tests field dependence under a column-permutation null; it does not certify every group or link. Native recursive selection and conditional certification use distinct codes. Repeated-prefix tests need a declared schedule. Fresh-block evidence supplies a separate global-null alarm. Conventional log-loss is an empirical prediction measure on a fixed vocabulary, not another form of that certificate.

11 Conclusion

ESCROW learns and evaluates organizations of arriving categorical records. Its complete description pays for overlapping memberships and cell ownership; its semantic head shares sparse counts; its memory head adapts trust in predictors. Selection extracts exact shallow hierarchies from external proposals, while pruning can remove reference groups at greater depths. Recursive search adds near-perfect two-level recovery, and selected parent-based insertion gives strong Amazon prediction. All selected methods select no groups on the matched null streams. Native and selected outcomes show that recovery and the two prediction measures can move in different directions.

AI USE STATEMENT

We used generative AI tools for three purposes: generating synthetic data sets, drafting sections of this paper, and aiding and polishing the writing. The synthetic depth and noise streams are produced by the seeded generators in the accompanying source; the seeds, generator source, and resulting inputs ship with their hashes. We have not used generative AI tools to develop the research idea, the method, or the theory, to formulate mathematical claims, to design the experiments, or to interpret the results. Those are the authors' own work.

We have reviewed all AI-assisted work. Every empirical claim rests on a recorded execution with preserved inputs, settings, prompts, and responses, and the accompanying source includes independent scoring checks, implementation comparisons, and replay results. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The evaluation uses two public corpora. Amazon Berkeley Objects contains product listings and no personal data; it is distributed under CC BY 4.0, with the data provided by Amazon.com and the dataset described by Collins et al. (2022). The Wikidata subset describes people and therefore contains personal data about identifiable individuals, taken from already public records released under CC0 (Vrandečić & Krötzsch, 2014). We use those records only as categorical field-value inputs. We release the derived record indices, counts, and hashes needed to reproduce every number, not enriched profiles, and no individual is named in the paper. Field and value names in the synthetic streams are hashed.

Because the method groups records by their fields, applying it to personal records can produce groups that correlate with sensitive attributes such as occupation or citizenship. We report recovered structure only in aggregate, and we make no claim that any recovered group is an appropriate category for people. The reference hierarchies are evaluation targets drawn from the corpora, not endorsements of those classifications. No human subjects were involved and no institutional review was required.

REPRODUCIBILITY STATEMENT

The appendix specifies the code, all theorem assumptions and proofs, experiment protocols, complete comparison tables, and language-model prompts. The accompanying source includes the data used to make every plot and table, a build script, and source hashes. Result families distinguish the fixed prediction pipeline from search and head component studies. The appendix maps each study to its execution path, memory scope, and protocol; a same-input code audit compares the two hierarchy proposers. Appendix A maps every reported result to the file in the supplementary archive that holds it, and gives the one command that rebuilds every figure and table from those files. The anonymised code, frozen evidence, build scripts, and prompts are also browsable at `https://anonymous.4open.science/r/ESCROW-9723/`.

REFERENCES

Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 7819 of *Lecture Notes in Computer Science*, pp. 160–172. Springer, 2013. doi: 10.1007/978-3-642-37456-2_14.

Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*, pp. 328–339. Society for Industrial and Applied Mathematics, 2006. doi: 10.1137/1.9781611972764.29.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. ABO: Dataset and benchmarks for real-world 3D object understanding. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, pp. 21126–21136, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Collins_ABO_Dataset_and_Benchmarks_for_Real-World_3D_Object_Understanding_CVPR_2022_paper.html.

Matteo Dell'Amico. FISHDBC: Flexible, incremental, scalable, hierarchical density-based clustering for arbitrary data and distance. *arXiv preprint arXiv:1910.07283*, 2019. URL https://arxiv.org/abs/1910.07283.

Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975. doi: 10.1109/TIT.1975.1055349.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987. doi: 10.1023/A:1022852608280.

Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, Cambridge, MA, 2007. ISBN 9780262072816. URL https://mitpress.mit.edu/9780262072816/the-minimum-description-length-principle/.

Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2): 151–178, 1998. doi: 10.1023/A:1007424614876.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255–264, 2017. doi: 10.1145/3097983.3098079.

R. Krichevsky and V. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, 1981. doi: 10.1109/TIT.1981.1056331.

Brian Kulis and Michael I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012. URL https://icml.cc/2012/papers/291.pdf.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205.

Nicholas Monath, Ari Kobren, Akshay Krishnamurthy, Michael R. Glass, and Andrew McCallum. Scalable hierarchical clustering with tree grafting. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1438–1448, 2019. doi: 10.1145/3292500.3330929.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://jmlr.org/papers/v12/pedregosa11a.html.

Tiago P. Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014. doi: 10.1103/PhysRevX.4.011047.

Tiago P. Peixoto. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1):012317, 2017. doi: 10.1103/PhysRevE.95.012317.

Dan Pelleg and Andrew W. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pp. 727–734. Morgan Kaufmann, 2000. URL https://www.cs.cmu.edu/~dpelleg/download/xmeans.pdf.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023. doi: 10.1214/23-STS894.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi: 10.1016/0377-0427(87)90125-7.

Johannes Ruf, Martin Larsson, Wouter M. Koolen, and Aaditya Ramdas. A composite generalization of Ville's martingale theorem using e-processes. *Electronic Journal of Probability*, 28:1–21, 2023. doi: 10.1214/23-EJP1019. Article 127.

Boris Ryabko and Jaakko Astola. Universal codes as a basis for time series testing. *Statistical Methodology*, 3(4):375–397, 2006. doi: 10.1016/j.stamet.2005.10.004.

Rylan Schaeffer, Yilun Du, Gabrielle K. Liu, and Ila Fiete. Streaming inference for infinite feature models. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19366–19387, 2022. URL https://proceedings.mlr.press/v162/schaeffer22a.html.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. doi: 10.1214/aos/1176344136.

Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: A nonparametric framework for sequential change detection. *The New England Journal of Statistics in Data Science*, 2(2): 229–260, 2024. doi: 10.51387/23-NEJSDS51.

Koen Smets and Jilles Vreeken. Slim: Directly mining descriptive patterns. In *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*, pp. 236–247. Society for Industrial and Applied Mathematics, 2012. doi: 10.1137/1.9781611972825.21.

Jean Ville. *Étude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939. URL https://www.numdam.org/item/THESE_1939__218__1_0/.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489.

Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011. doi: 10.1007/s10618-010-0202-x.

Christopher S. Wallace and David M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968. doi: 10.1093/comjnl/11.2.185.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 103–114, 1996. doi: 10.1145/233269.233324.

A WHERE TO FIND EACH RESULT IN THE SUPPLEMENTARY MATERIAL

The supplementary archive contains the method and experiment code, the frozen results behind every figure and table, the builders that regenerate them, and the dataset provenance. Its layout is below, followed by a map from each reported result to the file that holds it. The code and evidence are also browsable at https://anonymous.4open.science/r/ESCROW-9723/.

Table 3: Supplementary archive layout. Running `rebuild.sh` regenerates every figure and table from `evidence/` alone. The archive is the code artifact and does not restate the manuscript sources. It makes no model call and reruns no experiment.

Path	Contents
`code/escrow`	the method: coding, certification, search, prediction heads
`code/experiments`	experiment runners
`code/tests`	unit and property tests
`scripts/`	analysis that regenerates every figure and table from `evidence/`
`evidence/`	frozen results, protocols, audits, and all language-model logs
`prompts/`	exact language-model prompts and request builders
`source_snapshot/code`	the code state the recorded provenance hashes refer to
`data/`	dataset provenance, construction steps, and licences
`rebuild.sh`	one command that regenerates the figures and tables

Table 4: Map from each reported result to its evidence file. The matched report directory is `evidence/standard_baselines/derived/matched_prediction/report/`; "report/" below refers to this directory. Independent checks use `scripts/verify_current.py`, `scripts/verify_pruning.py`, and `scripts/audit_current_predictor.py`.

Reported result	File
Matched recovery and prediction for every proposer, Table 2 and Figures 3 and 4	`report/rows.json`
Paired order intervals and per-head means	`report/paired_intervals.json`, `report/head_intervals.json`
Current-system benchmark, 175 cases	`evidence/current_benchmark/results.json`
Pruning follow-up and its measurement audit	`evidence/pruning_comparison/`
Recursive versus parent-based same-input audit	`evidence/depth_code_audit/`
Matched COBWEB depth comparison	`evidence/cobweb_depth/depth_rows.json`
Classical noise ladder, corrected DP-means and BIRCH	`evidence/standard_baselines/`
Semantic and memory head component study	`evidence/head_components/`
Fresh-block evidence process	`evidence/wealth.json`
Every language-model request, response, and parser replay	`evidence/matched_prediction/llm/`
Language-model coverage and certification diagnostics	`report/llm_diagnostics.json`
Exact prompts and response contracts	`evidence/matched_prediction/prompts/`
Per-study protocols, settings, and input hashes	`evidence/protocols/`

The archive carries the results the paper reports and the records needed to re-derive and re-check them, not every raw byte. Superseded campaign dumps, per-case graph and diagnostic dumps whose aggregated outcomes are included, and code that no reported result uses are left out. Nothing reported here depends on an omitted file, and the three builders were run from a clean extraction of the archive to confirm it.

B THEORY, ASSUMPTIONS, AND COMPLETE PROOFS

This appendix separates the validity of a description, a test at one prefix, and a process that accumulates evidence on fresh blocks. All logarithms are base two unless written $\ln$. Lengths are ideal information lengths, so they can be fractional. A physical binary encoder can round a complete length upward; that rounding never improves a certificate. The conditional implementation evaluates lengths. It is not claimed to emit a byte stream of exactly that fractional length.

B.1 THE SAMPLE SPACE AND ITS REFERENCE DISTRIBUTION

Fix a finite set of fields A, a finite vocabulary for each field, and a prefix length n. Include absence $\perp$ in each vocabulary. The public header specifies those items. For a count table $t = (c_{av})$, define

$$C(t) = \{x : \#\{i : x_{ia} = v\} = c_{av} \text{ for every } a, v\}.$$

Each column has $n!/\prod_v c_{av}!$ distinct arrangements. Columns can be arranged independently, giving Eq. 1. The conditional reference is $U_t(x) = 1/N(t)$ for $x \in C(t)$. Conditioning removes the need to estimate marginal frequencies for the null test. It does not make those frequencies freely available to an unconditional predictor.

Lemma 1 (Uniformity on a column-permutation class)**.** *If the law of X is invariant under independent permutations of its columns, then $P(X = x \mid T = t) = 1/N(t)$ whenever $P(T = t) > 0$ and $x \in C(t)$.*

Proof. For any $x, y \in C(t)$, a permutation of each column sends x to y. Invariance gives $P(X = x) = P(X = y)$. The finite set $C(t)$ partitions the event $T = t$, so these equal conditional probabilities sum to one. They must each be $1/N(t)$. □

Independent and identically distributed records whose fields are independent satisfy this invariance, including unknown marginal distributions. The null also contains mixtures of such laws when the required invariance is retained. It is defined by invariance, not by a claim that every possible source of noise is independent.

B.2 THE COMPLETE CONDITIONAL DESCRIPTION

A description specifies a directed acyclic graph, a field support for each node, memberships, an owner for each cell, and the cell symbols. Nodes are in a deterministic parent-before-child order. Members of a child must also be members of every parent. The background, indexed by zero, is always an eligible owner. A non-background owner must contain the record and support its field. Canonical ordering removes arbitrary naming choices. Restrictions such as acyclicity and canonical ordering only discard otherwise available descriptions.

Primitive codes. A nonnegative integer m uses the gamma code of Elias (1975) for $m + 1$, of length $2\lfloor \log_2(m + 1) \rfloor + 1$. For a categorical sequence over a known d-symbol alphabet, with counts $c_1, \ldots, c_d$ and total m, the Krichevsky–Trofimov (Dirichlet-half) sequence code (Krichevsky & Trofimov, 1981) is

$$L_{\mathrm{KT}}(c) = \log_2 \frac{\Gamma(m + d/2)}{\Gamma(d/2)} - \sum_{j=1}^{d} \log_2 \frac{\Gamma(c_j + 1/2)}{\Gamma(1/2)}. \tag{6}$$

This is the negative log probability of the ordered sequence, not of its histogram. It equals the product of successive probabilities $(c_j^{\mathrm{past}} + 1/2)/(m^{\mathrm{past}} + d/2)$. For $d = 1$ or an empty sequence the charge is zero. A binary vector uses the same code with $d = 2$. Thus an edge or support vector is not silently charged only by its number of ones.

Graph and supports. Encode the node count K. Given K, encode the upper-triangular edge vector in the parent-before-child order, with $K(K - 1)/2$ slots. For each node encode its binary support vector over A. Invalid graphs are unused codewords. The one-bit code for $K = 0$ is why the empty graph has the reference length plus one bit.

Memberships. Collect nodes having exactly the same parent set into a family. Its eligible records are the intersection of those parents' member sets, or all records for a root family. For a family of j children, a record has a j-bit membership pattern. Encode the number of distinct patterns, encode each distinct pattern as a binary vector in canonical order, and encode the ordered sequence of pattern indices using Eq. 6. This represents overlap directly. It neither assumes disjoint children nor charges the same membership decision independently for every ancestor.

Owners. Given memberships and supports, each cell has a known list of eligible owners. For each field, collect records with the same eligible-owner list. Encode the owner sequence in that collection with Eq. 6, retaining zero-count eligible options in its alphabet. The implemented routing policy is deterministic given the graph, but the declared code still pays this owner charge. This is conservative and prevents a search policy from silently providing a free data-dependent route.

Owned cells. For each supported node and field, first encode presence among that node's owned records using the binary Dirichlet-half code. Encode the ordered present values using Eq. 6 on the public vocabulary. Counts are taken from owned cells, not all members. Otherwise an overlapping node could borrow a saving on a cell already encoded by another node.

The optional value code chooses between the Dirichlet-half code and an escape code, with a one-bit flag:

$$L_{\text{value}} = 1 + \min(L_{\text{KT}}, L_{\text{escape}}). \tag{7}$$

For alphabet size d, total count m, and k distinct observed values, the escape code with $\alpha = 1/2$ is

$$L_{\text{escape}} = \log_2 \frac{\Gamma(m+\alpha)}{\Gamma(\alpha)} - k \log_2 \alpha - \sum_{v:c_v>0} \log_2 \Gamma(c_v) + \log_2 \frac{d!}{(d-k)!}. \tag{8}$$

A previously seen value has probability $c_v/(m+\alpha)$; an unseen value has $\alpha/((m+\alpha)(d-k))$. When no unseen values remain, unused escape mass is allowed to terminate outside the valid sequence space. The code is then a subprobability code. The implementation uses zero bits when $d \le 1$.

Background cells. After decoding all owned cells, subtract their symbol counts from t. Negative remaining counts make a description invalid. For each field, encode its remaining ordered symbols by an index among all permutations of that remaining multiset. Its ideal length is the log multinomial count. This includes remaining absences. The original totals are side information for this conditional code, not a model estimated again separately for each candidate.

Lemma 2 (Composition and the Kraft bound)**.** *For every fixed header and count table t, the described construction obeys $\sum_D \sum_{x \in C(t)} 2^{-L_c(D,x|t)} \le 1$.*

Proof. The integer code has Kraft sum at most one. At every subsequent stage, the number of symbols and their alphabet are known from the header or the decoded prefix. The Dirichlet-half predictive probabilities sum to one at each step. The escape probabilities sum to at most one. For the flagged minimum code, $2^{-L_{\text{value}}} = \frac{1}{2}\max(2^{-L_{\text{KT}}}, 2^{-L_{\text{escape}}})$ is at most $\frac{1}{2}(2^{-L_{\text{KT}}} + 2^{-L_{\text{escape}}})$, so its sequence mass is at most one. The background multiset index is uniform on a finite set and has total mass one. Successively summing the product of these conditional masses from the last stage back to the first gives a total at most one. Invalid graphs, inconsistent counts, noncanonical representations, and invalid owner routes remove mass. Restricting the output to $C(t)$ also cannot increase it. The stages reconstruct D and every cell, so distinct valid described outcomes are counted within this bound. □

Decoder order. Read the public header and t; read nodes, edges, and supports; decode the families' memberships in parent order; decode eligible owner sequences; decode owned presence and values; then decode each remaining column arrangement. Check memberships, ownership, and totals. No choice in this sequence requires an undisclosed model selected from the data. The proof concerns this complete construction. A search objective with uncharged fitted parameters does not inherit it.

B.3 Proof of the prefix test and its scope

Proof of Theorem 1. Fix t. If a pair (D, x) clears the margin, then $L_c(D, x \mid t) \le \log_2 N(t) - b$, so $2^{-L_c(D,x|t)} \ge 2^b/N(t)$. Lemma 2 therefore allows at most $N(t)2^{-b}$ such pairs. There can be no more accepted arrays than accepted pairs. By Lemma 1, the conditional rejection probability is at most $N(t)2^{-b}/N(t) = 2^{-b}$. Average over T to obtain the claim. Searching any subset of descriptions, using any data-dependent procedure, cannot enlarge the event beyond the event in the theorem. □

The risk interpretation is one chance in at most $2^{10} = 1{,}024$ at one fixed prefix under the declared null. It is not a posterior probability that the selected graph is true. It is not a separate error probability for every node.

Proposition 1 (Why the conditional class matters)**.** *For the full family of column-permutation-invariant laws on this finite sample space, a rejection set R has size at most α for every null law if and only if $|R \cap C(t)|/N(t) \le \alpha$ for every t.*

Proof. The reverse implication follows by conditioning on T and averaging. For the forward implication, choose the null law uniform on any one $C(t)$. Under that law the rejection probability is exactly $|R \cap C(t)|/N(t)$. □

This establishes a conditional requirement for this broad null family. It does not say that ESCROW's code is the only valid test.

Repeated overlapping-prefix checks need a correction. One simple option tests prefix $j \ge 1$ with margin $b_j = b + \log_2(j(j+1))$. The union bound gives

$$\mathbb{P}\{\text{any false prefix alarm}\} \le \sum_{j=1}^{\infty} 2^{-b_j} = 2^{-b},$$

because $\sum_{j \ge 1} 1/(j(j+1)) = 1$. No independence between these checks is needed. The reported fixed-margin repair decisions do not use this correction. The fresh-block process below controls a separate evidence alarm; it does not supply a repeated-prefix bound for those repair decisions.

B.4 Fresh blocks and the urn code

Let $\mathcal{F}_{k-1}$ contain everything available before block k. Its model is measurable from that past. The block counts T_k become available for conditional assessment. We assume conditional uniformity of the block given $(\mathcal{F}_{k-1}, T_k)$. Independent records with independent fields and predetermined block boundaries satisfy it. A stopping rule that chooses the end of a block after inspecting its contents needs a new proof; it is not covered merely by calling the result a block.

Lemma 3 (Urn normalization)**.** *For any strictly positive predictable weights $M_{ia}(v)$, Eq. 5 defines a probability distribution on $C(T_k)$, including when the weights depend on cells already decoded within that block. With all weights one, it equals U_{T_k}.*

Proof. At each step, remaining counts are nonnegative and at least one is positive. The denominator is positive, the probabilities sum to one, and values with count zero cannot be drawn. Each draw decreases a remaining count by one. Thus all and only allowed arrangements are leaves of a finite probability tree of total mass one. When $M = 1$, a column sequence has probability $\prod_v c_{av}!/n!$ by cancellation of successive numerators and denominators. Multiplying over columns gives $1/N(T_k)$. □

Proof of Theorem 2. Write $E_k = 2^{\Delta_k} = N(T_k)Q_k(X_k \mid T_k)$. Conditional on the past and T_k,

$$\mathbb{E}[E_k \mid \mathcal{F}_{k-1}, T_k] = \sum_{x \in C(T_k)} \frac{1}{N(T_k)} N(T_k)Q_k(x \mid T_k) = 1.$$

The equality becomes an inequality for a subprobability code. Averaging over T_k and multiplying by the past-measurable W_{k-1} proves $\mathbb{E}[W_k \mid \mathcal{F}_{k-1}] \le W_{k-1}$. Nonnegativity and $W_0 = 1$ then imply

$\mathbb{P}(\sup_k W_k \geq 2^b) \leq 2^{-b}$ by Ville's inequality. For completeness, let τ be the first crossing. For any finite m, optional stopping at $\tau \wedge m$ gives $1 \geq \mathbb{E}[W_{\tau \wedge m}] \geq 2^b \mathbb{P}(\tau \leq m)$. Let m increase to infinity to obtain the bound. This proof permits past-based changes of graph and predictor between blocks. It provides evidence against the global null; it does not validate later refinements individually. □

B.5 Expert mixtures and their actual cost

Each expert supplies a normalized next-cell probability. With weights w_{tj}, the mixture is $q_t = \sum_j w_{tj} q_{tj}$. After a cell z_t, a Bayesian update is $\widetilde{w}_{t+1,j} = w_{tj} q_{tj}(z_t)/q_t(z_t)$. Fixed sharing at rate ρ sets $w_{t+1,j} = (1-\rho)\widetilde{w}_{t+1,j} + \rho/J$. Both updates preserve normalization and hence Lemma 2's block analogue.

Proposition 2 (Mixture comparison)**.** *With $\rho = 0$ and initial weights π_j, a mixture's sequence length is at most $L_j - \log_2 \pi_j$ for every fixed expert j. With fixed sharing, comparison to an expert path $(j_1, \ldots, j_m)$ pays instead*

$$-\log_2 \pi_{j_1} - \sum_{t=2}^{m} \log_2 \big((1-\rho)\mathbf{1}\{j_t = j_{t-1}\} + \rho/J\big).$$

Proof. For zero sharing, the sequence probability equals $\sum_j \pi_j \prod_t q_{tj}(z_t)$ and is at least the term for expert j. With sharing, expand the same recursion as a sum over paths of a Markov chain with transition probabilities $(1-\rho)\mathbf{1}\{j' = j\} + \rho/J$. Keep one path and take negative logarithms. □

Thus a two-expert uniform mixture without sharing pays at most one bit per independently initialized mixture. At positive sharing, even a fixed expert pays $\log_2 J + (m-1)[-\log_2(1-\rho+\rho/J)]$. A universal one-bit claim would be incorrect for the later shared mixture. The code initializes separate expert weights by field; the full charge sums over those initializations.

B.6 Population explanation for support and prediction

Let $(G, X_1, \ldots, X_p)$ have finite alphabets. Assume the fields are independent given G. A hard description that names G and then describes the fields has population rate $H(G) + \sum_a H(X_a \mid G)$. The independent-field reference rate is $\sum_a H(X_a)$. Subtracting gives

$$R_{\text{hard}} = \sum_a I(X_a; G) - H(G). \tag{9}$$

A correct mixture describes X without naming its component. Its rate is $H(X) = H(X \mid G) + I(X; G)$, yielding

$$R_{\text{soft}} = \sum_a I(X_a; G) - I(X; G) = \text{TC}(X), \qquad R_{\text{soft}} - R_{\text{hard}} = H(G \mid X). \tag{10}$$

These are identities obtained from $I(X_a; G) = H(X_a) - H(X_a \mid G)$ and $H(G) - I(X; G) = H(G \mid X)$. Conditional independence supplies $H(X \mid G) = \sum_a H(X_a \mid G)$.

For iid samples, consistent component coding, correct component assignments in the hard comparison, and sublinear overhead, the corresponding normalized lengths converge to these rates. A strictly positive limiting saving eventually exceeds any fixed margin when those convergence assumptions hold. A zero or negative limit gives no such conclusion. Search may fail to supply the needed components, the component count may grow, and the real block scorer conditions on counts. The identities explain a mechanism; they do not prove universal recovery or equate its finite-block score with unconditional log-loss.

Finally, Kraft's inequality alone does not prohibit a singleton proposal. A code could assign high prior mass to a specified singleton. The implementation's minimum candidate size and the charges for naming a group are concrete design choices, not a universal impossibility theorem.

C IMPLEMENTATION AND INFORMATION ACCESS

C.1 COMPLETE PREDICTION PROCEDURE

The following procedure specifies the support and prediction interface used by the system. The matched three-block study holds the fitted graph, counts, and similarities fixed across its evaluation blocks. The sequential evidence study rebuilds between blocks with its separately declared two-expert configuration.

Table 5: The prefix decision and the accumulated-evidence alarm are separate outputs. The matched benchmark uses the fixed-model branch of step 7.

Algorithm 1: propose, certify, and evaluate

Input: categorical records; block boundaries; repair cadence; margin b.
State: prefix, candidate tree, certified graph, value similarities, expert weights, $\log_2 W = 0$.
1. Use the available prefix to build or update candidate groups.
2. Select fields, ownership, and groups using the conditional certifier.
3. If its prefix score is below b, use the empty selected graph.
4. Fit group counts and the semantic head from the prefix; freeze both.
5. Score the next block using its counts; the memory head updates after each cell.
6. Add Δ_k to $\log_2 W$; report any global-null threshold crossing.
7. Repeat steps 5–6 with the fixed model, or update the prefix and restart at step 1.

C.2 RECURSIVE HIERARCHY SEARCH

The recovery engine is `FastCachedBoundaryCascadeGraph`, with recursive local workers, a repair every 100 records, a candidate budget of 4,096 evaluations per repair, and at most 16 candidate components. It performs one declared terminal repair. Local windows have depth at most three, but nested modules can produce a deeper global graph. Every proposed graft is validated against the outside graph. An edit is accepted only when the complete EHR05 ideal description and its serialized description improve. This native rule is a complete-code comparison, not the EHR07 conditional certificate.

The search can propose memberships, supports, and parent links through repeated repairs. It is therefore different from the insertion-only tree below. The common conditional certifier can also evaluate its returned memberships and links. Appendix E.9 keeps the native graph, whole-certified graph, and pruned graph separate, with identical inputs and recovery metrics. The theoretical prefix guarantee applies to the conditional gate when used; it is not retroactively assigned to the native EHR05 search.

C.3 PARENT-BASED INSERTION PROPOSER

The parent-based tree starts with $\alpha = 2$, child-creation mass $\theta = 1$, and minimum exported group size two. A new record updates the root, then descends. At each node, compare each existing child with a fresh child. An existing child pays $-\log_2(n_c/(\sum_j n_j + \theta))$ to be chosen; a fresh child pays $-\log_2(\theta/(\sum_j n_j + \theta))$. Add its predictive length for all fields. Choose the least costly option; a tie with a fresh child creates that child.

For a present value v of field a, the child probability is $(c_{av} + \alpha_a p_{\text{parent}}(v))/(m_a + \alpha_a)$, where m_a counts present observations. Presence itself is smoothed toward the parent. The smoothing grid is $\{.25, 1, 4, 16, 64, 256, 1024, 4096, 16384, 65536\}$. The implementation refits at scheduled count thresholds, then sets the next threshold to $\max(4, 2m_a)$. These fitted proposal scores are not the final conditional certificate.

The parent-based insertion method scores every field with the child's own smoothed counts. It does not use the experimental per-field parent-deference rule or invoke the separately defined child-merge routine. This distinction matters: irrelevant fields can still fragment the proposal tree. Exported groups have at least two and fewer than all prefix records. Parent relations for the exported pool are reconstructed from minimal strict set inclusion.

C.4 SEARCH AND CERTIFICATION

The selected arms of all real-data proposers enter the same fast certifier with the optional flagged value code. It installs memberships consistently with parents, removes transitive edges, and begins with empty field supports. For each field it adds the support with greatest positive saving, with ownership routed in deterministic parent-before-child order. It then deletes costly nodes, reconnects their children to their parents, and revisits affected supports. Candidate gains in the lazy queues are recomputed before use and before declaring that no improvement remains.

The search reference uses frozen whole-column probabilities for background cells. The final reference instead uses the remaining column-type counts. Only the latter establishes the stated certificate. The resulting graph is kept only if its final score clears the margin. The fast version also refuses node removals that would leave a separately read-out owned group unnamed in the structure charge. The supplied source snapshot is authoritative for pass order and tie behavior. This is a heuristic search, with no claim of global optimality.

C.5 THE BLOCK PREDICTOR AND ITS TWO HEADS

The predictor builds components from the selected groups plus the background. It fits every field in each group; selected supports and ownership do not gate these predictive counts. Those choices affect certification, while the group readout determines the predictor's components. It uses group counts with half-count smoothing. Component weights are proportional to component size plus $1/2$, normalized by the sum of component sizes plus half the number of components. Because memberships can overlap, that denominator need not equal the number of records. Within a record, component weights update using fields already encoded. A past-fixed graph and prefix-derived parameters provide the positive weights in Eq. 5.

The conditional real-data comparison uses three next-cell experts: group prediction, the untilted urn, and the semantic head's group predictor. The memory head maintains one weight vector per field and updates it after each observed cell, with sharing rate $\rho = .01$. The experiments reset these vectors at each block. T9 uses only the first two experts with $\rho = 0$, also reset at each block. These are explicit component configurations of the block predictor. Appendix D defines the separately normalized conventional predictor, whose reference and vocabulary depend only on the training prefix.

Semantic head. For a field, represent each value by frequencies of its co-occurring other field–value pairs in the prefix. Let s_{uv} be cosine similarity clipped to $[0, 1]$. Set $S_{uv} = \exp((s_{uv} - 1)/.25)$ and $S_{uu} = 1$. Replace a component's count vector c by Sc when making its shared-count expert. This is applied only to eligible vocabularies with two through 4,000 values. Sharing happens inside each component, not by assigning a value to a group from held-out labels.

Memory head. Let q_{tj} be expert j's probability for the next cell in one field, and let w_{tj} be its current weight. The head predicts $q_t = \sum_j w_{tj} q_{tj}$. After seeing value z_t, it updates

$$\widetilde{w}_{t+1,j} = \frac{w_{tj} q_{tj}(z_t)}{q_t(z_t)}, \qquad w_{t+1,j} = (1 - \rho)\widetilde{w}_{t+1,j} + \rho/J.$$

Each predictor receives at least ρ/J weight after sharing. This lets a useful predictor regain influence after early losses. This is a uniform-sharing variant of fixed-share (Herbster & Warmuth, 1998). Our update shares mass among all J experts, including the source expert. It equals their update that shares only with other experts when $\alpha = \rho(J - 1)/J$. The sharing rate is fixed, not learned. All weights are computed before the next cell is scored. The mixture remains normalized; Appendix B proves the comparison to fixed experts and switching expert paths, including the actual sharing cost.

The implementation stores these weights as log weights indexed by field. Passing a persistent `state` dictionary to `block_delta` carries them across blocks; omitting it initializes uniform weights for each block. The R27 component study, LLM comparisons, and matched three-block benchmark omit it. The accumulated log-evidence is a separate scalar outside this function. Resetting predictor weights does not reset that evidence. Persistent three-expert performance is not inferred from these block-reset results.

Execution map. The matched pipeline begins with either recursive search or `ParentCentred`. Its selected arm then calls `FastCertifier`; its native arm retains the proposer's graph. Both feed the same `MixtureModel`, `co_occurrence_kernels`, `smooth_per_node`, and prediction scorers. The semantic kernel and group counts use only the prefix. The 82-input protocol in Appendix D evaluates both searches with this common interface. On synthetic inputs, recovery F1 measures the graph itself; neither prediction head enters that metric. The flat noise ladder separately evaluates its declared split search and does not supply a prediction result for either matched pipeline.

What the next block reveals. The block's field inventory and counts are used by the conditional scorer for every arm. Kernel similarities are built from the prefix. No arm learns group memberships from the evaluation block before that block is scored. Calling the result unconditional next-record prediction would nevertheless be inaccurate: the evaluator knows the block's marginal counts. Appendix E.6's ordinary prediction study answers a separate question without this reference conditioning.

C.6 Numerical and computational details

Multinomial terms are evaluated using log-gamma functions. Arrays and cached terms speed the same objective. Predictive probabilities use floating-point arithmetic and numerical floors to avoid evaluating $\log 0$. The theorems apply to the ideal normalized probabilities. Tests compare the array and reference lengths on small cases and enumerate small Kraft sums; these checks support the implementation but are not a formal proof of floating-point behavior for all inputs. Runtime comparisons use their declared implementations and saved measurements. The pruning study records its own timings, which are not used as a hardware-controlled speed comparison.

D MATCHED STRUCTURE, PREDICTION, AND WIKIDATA COMPARISON

D.1 QUESTIONS AND INFORMATION AVAILABLE TO EACH METHOD

This study connects the structure and prediction measurements. Its first question is whether the graphs returned by recursive search and parent-based insertion differ in their ability to predict later records when selection and prediction are held fixed. Its second question is where LLM and classical proposals stand on the same Wikidata records. Its third question is whether adaptive expert weights help beyond a fixed mixture of the same experts. The protocols and baseline configurations are recorded with their outcomes. Inputs and graph outputs are preserved. Repeated measurements on the same inputs are paired comparisons, not independent replications.

All 42 depth cases use the archived 300-record prefix and its following 100 records. The four proposers are recursive ESCROW, parent-based ESCROW, COBWEB, and FISHDBC. All 40 real cases use the same sampled orders as Appendix E.8: four feature variants, five orders per variant, and signal or column-shuffled control. Each case has a 600-record prefix and three following blocks of 100. Structure and component counts are fitted once to the prefix. No proposer sees a later block, reference group, reference link, target depth, or true group count.

The two ESCROW proposers use their declared implementations and budgets. Recursive search uses a 4,096-candidate budget, at most 16 components per local search, repairs every 100 arrivals, and one final repair. Its native EHR05 rule remains part of search. Native output and output after conditional pruning are reported separately for every proposer. Conditional selection uses the same `FastCertifier`, minimum value code, and margin $b = 10$. The selected explicit nodes define prediction components; the residual readout is not substituted.

D.2 CLASSICAL METHODS ON WIKIDATA

Both Wikidata variants and all their controls receive the same additional baseline set. Agglomerative clustering, leader clustering, and HDBSCAN use the transferred settings .45, .92, and 5. DP-means and BIRCH use $\lambda = .4945$ and radius .6705, selected on the separate synthetic development stream with the algorithms and grids in Appendix E.1. No setting is optimized on Wikidata. K-means chooses k from 2 through 20 by silhouette (Rousseeuw, 1987) on the training prefix, using ten initializations and random state zero. This choice uses no future score or labels. The normalized multi-hot input preserves field-value pairs and field presence. Batch methods can inspect the whole training prefix.

GRINCH uses the same normalized multi-hot vectors, converted to float32, and exports internal nodes with at least five and fewer than all prefix records. Nested SBM uses the hierarchical, microcanonical block model (Peixoto, 2014; 2017). Our adapter constructs a record–field-value bipartite graph, keeps record and token vertices in distinct classes, and exports nontrivial record groups from all hierarchy levels. Both use the input's declared seed and the preserved baseline implementations.

Noise labels are background. The shared adapter excludes singleton and universal groups, then records native and conditionally pruned output. GRINCH retains its five-member minimum. The two-member adapter rule for the other methods differs from the five-member eligibility rule in the separate classical noise-grid study. The full set is displayed together, without selecting a favorable parameter after examining the test results. True-k and oracle settings have no real-data ground truth available and are not treated as deployable Wikidata baselines.

D.3 A CONVENTIONAL PREDICTIVE MEASURE

Conditional bits saved measures how well a model explains arrangements given a block's column counts. We also report a separate, conventional chain-rule log-loss whose predictions do not use those counts. The training prefix fixes the field order and each field's vocabulary. The order is lexicographic. A field vocabulary contains the observed values, an absent symbol, and one unknown-value symbol. An unseen test value maps to the unknown symbol. Fields that first appear after training are excluded and counted explicitly. Every method receives the same vocabulary and is scored on the same cells.

For component g, field k, and vocabulary symbol v, ordinary probabilities are

$$p_g(v \mid k) = \frac{c_{gkv} + 1/2}{m_g + |V_k|/2}.$$

The component prior is proportional to $m_g + 1/2$. Component weights update after each observed field within a record, then reset for the next record. This defines a normalized joint distribution by the chain rule. Counts remain fixed at the training prefix. The reference uses the same smoothed whole-prefix marginal.

For the semantic expert, the prefix-only similarity kernel spreads counts within each component. We add the same half-count smoothing and normalize over the fixed vocabulary separately for each component. Absent counts are retained, and the unknown symbol receives its half-count. This is a normalized adaptation for conventional prediction; it is distinct from the conditional urn tilt. The memory head combines group, semantic, and reference probabilities and updates only after the cell being scored has been observed. Its sharing rate is .01, and weights reset at each block boundary.

The primary conventional number is total negative log probability in bits divided by the number of scored cells. We also retain the loss on present cells, the loss on seen symbols, the fraction of present values mapped to unknown, and the number of excluded new fields. This is loss on a declared mapped vocabulary. It does not measure the probability of spelling a previously unseen entity name. Reporting the unknown-value fraction prevents an apparent improvement from being mistaken for exact prediction of new categorical identities.

Nearest-neighbor prediction. The 10-nearest-neighbor predictor uses only the already observed earlier fields of the current record to retrieve training records. It uses mismatch distance and includes all ties at the tenth neighbor's distance. With no earlier fields it uses the full prefix. Half-count smoothing applies on the same fixed vocabulary. It never reads the target field while choosing neighbors. KNN is a prediction baseline; no hierarchy, certificate, or recovery score is attributed to it.

D.4 Separating semantic sharing from adaptive memory

Each graph is evaluated with five settings: group prediction alone; a fixed equal mixture of group and reference; adaptive weights over those two experts; a fixed equal mixture of group, semantic, and reference; and adaptive weights over those three experts. The last setting is the full predictor. Comparing the last two isolates adaptive weighting while retaining the same experts. Comparing the two adaptive settings measures adding the semantic expert. It does not isolate the sharing rate, forgetting, or persistence across blocks. All settings use the same graph, training records, field order, and evaluation cells.

D.5 LLM prompts, responses, and coverage

The Wikidata comparison uses zero-shot and few-shot proposals from Gemini and Qwen on the same prefixes as the non-LLM methods. Each request contains twenty new records and descriptions of the groups created so far. Few-shot requests prepend the three fixed bird examples printed in Appendix H. These give extra task guidance, with no examples from the evaluated domain. ESCROW learns from the observed records without supplied labels, domain instructions, pretrained encoders, or demonstrations. Its semantic similarities come from prefix counts. Thus ESCROW is zero-shot with respect to supplied domain knowledge throughout; the zero-shot and few-shot labels distinguish the LLM prompt conditions only. The models are asked to assign each current record to an existing group, create a group, or leave the record in background. These are group-proposal baselines evaluated with the shared predictor, not direct next-token likelihoods from the LLMs.

The parser accepts a JSON array, optionally inside a code fence. It checks one assignment per current record. Duplicate IDs, out-of-chunk IDs, missing responses, unknown groups, and malformed entries are counted separately. An invalid or missing assignment leaves that record in background. Malformed model output is not repaired or retried. Only declared transient transport failures can trigger a retry. Raw responses and every exact request are retained, including failed attempts.

Gemini uses the returned service identifier recorded by the API, temperature zero, low reasoning effort, and an 8,192-token output limit. Qwen2.5-14B-Instruct (Qwen Team, 2024) uses greedy

decoding, bfloat16 weights, and an 8,192-new-token limit. Its checkpoint revision, chat template, library versions, hardware, and batch size are recorded. Requests, responses, token counts, finish reasons, coverage, memberships, certificates, and future scores are linked by input and source hashes in the evidence package. The exact prompt templates remain in Appendix H; the supplementary request logs supply every concrete record-filled prompt.

Across 600 requests per prompt condition, valid assignment coverage is 99.43% for Gemini zero-shot, 99.08% for Gemini few-shot, 83.18% for Qwen zero-shot, and 72.12% for Qwen few-shot. Valid assignments include explicit background. Qwen's lower coverage is part of the observed interface performance, so its empty selected graphs cannot be attributed solely to an inability to find groups. Few-shot Gemini retains signal groups on nine of ten inputs. None of the four conditions retains a group on its ten null inputs. Table 12 separates parse errors, output limits, and assignment coverage.

D.6 Execution environment and computational cost

Qwen uses two NVIDIA A100 GPUs with 80 GB each, one independent shard per GPU, batch size four, and left padding. The recorded stack is PyTorch 2.5.1 with CUDA 12.1 and Transformers 5.16.1. The checkpoint revision and exact chat wrapper are included with the prompt settings. CPU proposal runs use eight independent workers with one numerical-library thread each, Python 3.12.3, NumPy 2.2.6, SciPy 1.15.3, and scikit-learn 1.7.2 (Pedregosa et al., 2011). The flat HDBSCAN arm uses `sklearn.cluster.HDBSCAN`; the separate `hdbscan` 0.8.40 package (McInnes et al., 2017) supplies hierarchy utilities used by FISHDBC. GRINCH uses version 0.2; nested SBM uses graph-tool 3.7 in its separate environment. DP-means and BIRCH use Python 3.12.12, NumPy 2.2.6, and scikit-learn 1.7.2 on CPU. Their source and environment records are included with their complete outcomes.

Nine recursive signal runs reached the original 1,800-second process allowance. All nine were rerun with a uniform 7,200-second allowance, with the algorithm, input, search budget, and repair schedule unchanged. All completed. The slowest recorded proposal took 4,923.8 seconds on a 600-record Wikidata prefix. The failed attempts remain in the evidence package. These costs show why reducing candidate generation and certification work is a practical research priority; they are not a hardware-controlled speed ranking across methods. Gemini records 8,579,901 tokens across completed calls; Qwen records 7,401,948. Transport-failure attempts are retained separately and may lack usage totals.

D.7 Measurement checks

An exhaustive small type class checks that the conditional probabilities sum to one for every head setting. An independent direct calculation checks the conventional group-mixture joint probability. Per-component probabilities sum to one over the fixed vocabulary. A one-field KNN check recovers the prefix marginal, as required when there is no retrieval context. Parser tests cover duplicate, missing, out-of-chunk, background, and unknown-group responses. Every full conditional score is checked against the unchanged scalar scorer. The evidence records any floating-point difference rather than claiming bitwise identity.

The complete audit verifies all 82 input hashes and 139 frozen source hashes, reconstructs 400 real-data certificates, and replays all 2,400 LLM requests and parser outputs. The maximum certificate discrepancy is 1.9×10^{-12} bits. Across every graph and future block, the maximum full-predictor discrepancy from the scalar implementation is 2.7×10^{-5} bits per block, below the declared 10^{-4} tolerance. Preserved baseline scores agree exactly with their saved values. All 568 declared proposer–input outcomes are present.

Independent adaptive-memory check. The conditional memory effect on Amazon was checked by a second implementation: the scalar scorer was evaluated with its expert updates disabled to obtain the fixed mixture. Twenty first-block checks cover both ESCROW searches and all ten Amazon feature-variant orders. Their maximum difference from the vectorized fixed mixture is 8.3×10^{-6} bits per block. This confirms that the conditional tradeoff is an outcome of adaptive weighting under this measure.

D.8 COMPLETE OUTCOMES AND PAIRED UNCERTAINTY

The tables report every method, both native and selected graphs, and every sampled order. Blocks are averaged within an order before comparing methods. Descriptive 95% intervals resample the five paired order differences with replacement 20,000 times, using random seed 92024. They describe variation on the fixed corpora and do not estimate variation over unseen domains. A positive difference favors the parent-based ESCROW pipeline. The conventional loss reduction reverses the sign because lower loss is better.

Table 6: Conventional loss in bits per scored cell, averaged across five orders. Graph methods use conditional selection and both prediction heads. Background uses the same heads with no groups. KNN is a separate prefix-only predictor.

Method	Amazon	Amazon*	Wikidata	Wikidata*
ESCROW recursive	2.060	1.899	0.902	0.924
ESCROW parent	2.056	1.879	0.911	0.933
COBWEB	2.086	1.952	0.936	0.968
FISHDBC	2.108	1.976	0.908	0.932
Background	2.382	2.122	0.943	0.973
10-NN	2.210	2.000	1.071	1.072

Table 7: Conditional bits saved per record for the same selected graphs. Zero denotes the untilted block-count reference. Background with the prediction heads can have a nonzero score. Conventional loss uses a different reference.

Method	Amazon	Amazon*	Wikidata	Wikidata*
ESCROW recursive	1.036	0.441	-0.629	-0.808
ESCROW parent	1.434	0.753	-0.845	-0.839
COBWEB	1.032	0.376	-1.371	-1.083
FISHDBC	0.705	0.093	-1.068	-1.010
Background	-0.264	-0.241	-1.527	-1.272

Effect of selection on parent-based prediction. Table 8 compares the same graph proposer and prediction heads before and after conditional selection. Conditional saving decreases on all four variants. Conventional loss improves on both Amazon variants and worsens on both Wikidata variants. Thus the direction of the prediction effect depends on the measure and corpus. These outcomes do not imply that selection always lowers prediction: Table 2 shows improved prediction for selected COBWEB on the depth streams.

Table 8: Parent-based ESCROW with semantic sharing and adaptive memory. Each mean averages three future blocks within each of five orders. Δ is selected minus native: positive is better for conditional saving and negative is better for conventional loss. * removes hierarchy-revealing fields.

	Conditional bits / record			Conventional bits / cell		
Corpus	Native	Selected	Δ	Native	Selected	Δ
Amazon	1.688	1.434	-0.255	2.112	2.056	-0.056
Amazon*	0.950	0.753	-0.197	1.930	1.879	-0.052
Wikidata	0.236	-0.845	-1.081	0.889	0.911	0.022
Wikidata*	-0.129	-0.839	-0.710	0.912	0.933	0.021

All Wikidata methods. The complete native and selected means are below. Asterisks indicate removal of hierarchy-revealing fields. All means use the same five orders and three future blocks per order.

Table 9: Wikidata signal means. N: native graph. S: conditional selection.

Method	N bits/record	S bits/record	N loss/cell	S loss/cell
ESCROW recursive	-0.629	-0.629	0.902	0.902
ESCROW parent	0.236	-0.845	0.889	0.911
COBWEB	-0.292	-1.371	0.938	0.936
FISHDBC	-0.865	-1.068	0.894	0.908
Agglomerative	-0.246	-1.396	0.937	0.934
DP-means	-1.358	-1.527	0.942	0.943
Leader	-0.481	-1.477	0.938	0.934
BIRCH	-0.331	-1.181	0.928	0.930
HDBSCAN	-1.236	-1.369	0.903	0.933
K-means (silhouette)	-0.538	-0.538	0.897	0.897
GRINCH	-0.119	-0.734	0.896	0.911
Nested SBM	-1.527	-1.527	0.943	0.943
Gemini zero-shot	-1.151	-1.527	0.938	0.943
Gemini few-shot	-0.451	-1.277	0.906	0.926
Qwen zero-shot	-1.348	-1.527	0.923	0.943
Qwen few-shot	-1.508	-1.527	0.932	0.943
Background	-1.527	-1.527	0.943	0.943
10-NN	–	–	1.071	1.071

Table 10: Wikidata* signal means, with the same methods and selection rules.

Method	N bits/record	S bits/record	N loss/cell	S loss/cell
ESCROW recursive	-0.754	-0.808	0.913	0.924
ESCROW parent	-0.129	-0.839	0.912	0.933
COBWEB	-0.556	-1.083	0.966	0.968
FISHDBC	-0.944	-1.010	0.930	0.932
Agglomerative	-0.383	-1.209	0.965	0.964
DP-means	-1.124	-1.272	0.971	0.973
Leader	-0.630	-1.225	0.964	0.972
BIRCH	-0.607	-0.981	0.950	0.961
HDBSCAN	-0.955	-1.125	0.930	0.960
K-means (silhouette)	-0.977	-0.977	0.925	0.925
GRINCH	-0.425	-0.983	0.920	0.948
Nested SBM	-1.238	-1.272	0.966	0.973
Gemini zero-shot	-1.033	-1.272	0.967	0.973
Gemini few-shot	-0.668	-1.060	0.943	0.954
Qwen zero-shot	-1.032	-1.272	0.948	0.973
Qwen few-shot	-1.225	-1.272	0.954	0.973
Background	-1.272	-1.272	0.973	0.973
10-NN	–	–	1.072	1.072

Paired differences. A: Amazon; W: Wikidata; *: removed fields. Brackets give descriptive 95% paired bootstrap intervals. Positive values favor parent-based ESCROW.

Data	Comparison	Pairs	Bits/record gain	Loss reduction
A	ESCROW recursive	5	0.398 [0.252, 0.535]	0.003 [-0.004, 0.015]
A	COBWEB	5	0.401 [0.296, 0.518]	0.030 [0.006, 0.054]
A	FISHDBC	5	0.729 [0.620, 0.848]	0.051 [0.030, 0.077]
A	Background	5	1.698 [1.536, 1.830]	0.326 [0.294, 0.354]
A	10-NN	5	–	0.154 [0.126, 0.179]
A*	ESCROW recursive	5	0.312 [0.125, 0.557]	0.020 [0.005, 0.033]
A*	COBWEB	5	0.377 [0.164, 0.652]	0.073 [0.026, 0.144]
A*	FISHDBC	5	0.660 [0.496, 0.824]	0.097 [0.070, 0.124]
A*	Background	5	0.994 [0.866, 1.159]	0.243 [0.220, 0.266]
A*	10-NN	5	–	0.122 [0.102, 0.141]
W	ESCROW recursive	5	-0.216 [-0.434, -0.007]	-0.009 [-0.015, -0.001]
W	COBWEB	5	0.525 [0.395, 0.710]	0.025 [0.018, 0.032]
W	FISHDBC	5	0.223 [0.170, 0.288]	-0.002 [-0.014, 0.005]
W	Agglomerative	5	0.551 [0.338, 0.765]	0.023 [0.016, 0.029]
W	DP-means	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	Leader	5	0.632 [0.406, 0.839]	0.023 [0.017, 0.029]
W	BIRCH	5	0.336 [0.198, 0.506]	0.019 [0.013, 0.025]
W	HDBSCAN	5	0.524 [0.339, 0.744]	0.022 [0.006, 0.036]
W	K-means (silhouette)	5	-0.308 [-0.808, 0.219]	-0.014 [-0.029, 0.008]
W	GRINCH	5	-0.111 [-0.359, 0.039]	0.000 [-0.015, 0.011]
W	Nested SBM	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	Gemini zero-shot	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	Gemini few-shot	5	0.432 [0.175, 0.688]	0.015 [0.010, 0.020]
W	Qwen zero-shot	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	Qwen few-shot	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	Background	5	0.682 [0.538, 0.832]	0.032 [0.026, 0.039]
W	10-NN	5	–	0.160 [0.151, 0.168]
W*	ESCROW recursive	5	-0.031 [-0.331, 0.269]	-0.009 [-0.033, 0.022]
W*	COBWEB	5	0.244 [-0.003, 0.599]	0.035 [0.025, 0.045]
W*	FISHDBC	5	0.172 [-0.124, 0.491]	-0.000 [-0.018, 0.016]
W*	Agglomerative	5	0.370 [0.159, 0.670]	0.031 [0.014, 0.050]
W*	DP-means	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	Leader	5	0.387 [0.199, 0.659]	0.039 [0.027, 0.052]
W*	BIRCH	5	0.143 [-0.108, 0.360]	0.028 [0.012, 0.040]
W*	HDBSCAN	5	0.286 [0.002, 0.622]	0.027 [0.016, 0.039]
W*	K-means (silhouette)	5	0.139 [-0.018, 0.374]	-0.008 [-0.021, 0.004]
W*	GRINCH	5	0.145 [-0.158, 0.390]	0.015 [-0.007, 0.035]
W*	Nested SBM	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	Gemini zero-shot	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	Gemini few-shot	5	0.221 [0.043, 0.377]	0.021 [0.001, 0.041]
W*	Qwen zero-shot	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	Qwen few-shot	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	Background	5	0.433 [0.246, 0.688]	0.040 [0.027, 0.053]
W*	10-NN	5	–	0.139 [0.132, 0.145]

Every prediction-head setting. The graph is fixed within each search and order. "Fixed" assigns equal weights to the same experts used by its adaptive counterpart. Group-only prediction removes both expert mixing and semantic sharing.

Data	Search	Prediction heads	Orders	Bits/record	Loss/cell
Amazon	recursive	Group only	5	-4.619	2.046
Amazon	recursive	Group + ref., fixed	5	0.755	2.143
Amazon	recursive	Group + ref., adaptive	5	0.877	2.053
Amazon	recursive	Three experts, fixed	5	1.456	2.199
Amazon	recursive	Three experts, adaptive	5	1.036	2.060
Amazon	parent	Group only	5	-3.060	2.045
Amazon	parent	Group + ref., fixed	5	1.047	2.129
Amazon	parent	Group + ref., adaptive	5	1.194	2.049
Amazon	parent	Three experts, fixed	5	1.786	2.178
Amazon	parent	Three experts, adaptive	5	1.434	2.056
Amazon*	recursive	Group only	5	-5.203	1.885
Amazon*	recursive	Group + ref., fixed	5	0.175	1.949
Amazon*	recursive	Group + ref., adaptive	5	0.433	1.892
Amazon*	recursive	Three experts, fixed	5	0.566	2.012
Amazon*	recursive	Three experts, adaptive	5	0.441	1.899
Amazon*	parent	Group only	5	-3.714	1.869
Amazon*	parent	Group + ref., fixed	5	0.375	1.923
Amazon*	parent	Group + ref., adaptive	5	0.634	1.872
Amazon*	parent	Three experts, fixed	5	1.016	1.972
Amazon*	parent	Three experts, adaptive	5	0.753	1.879
Wikidata	recursive	Group only	5	-47.323	0.900
Wikidata	recursive	Group + ref., fixed	5	-2.798	0.906
Wikidata	recursive	Group + ref., adaptive	5	-0.777	0.898
Wikidata	recursive	Three experts, fixed	5	-2.384	0.942
Wikidata	recursive	Three experts, adaptive	5	-0.629	0.902
Wikidata	parent	Group only	5	-40.512	0.912
Wikidata	parent	Group + ref., fixed	5	-2.973	0.911
Wikidata	parent	Group + ref., adaptive	5	-1.051	0.907
Wikidata	parent	Three experts, fixed	5	-2.070	0.945
Wikidata	parent	Three experts, adaptive	5	-0.845	0.911
Wikidata*	recursive	Group only	5	-54.107	0.919
Wikidata*	recursive	Group + ref., fixed	5	-3.611	0.933
Wikidata*	recursive	Group + ref., adaptive	5	-1.019	0.921
Wikidata*	recursive	Three experts, fixed	5	-2.693	0.964
Wikidata*	recursive	Three experts, adaptive	5	-0.808	0.924
Wikidata*	parent	Group only	5	-41.076	0.933
Wikidata*	parent	Group + ref., fixed	5	-3.048	0.937
Wikidata*	parent	Group + ref., adaptive	5	-0.991	0.930
Wikidata*	parent	Three experts, fixed	5	-2.297	0.971
Wikidata*	parent	Three experts, adaptive	5	-0.839	0.933

Table 11: Mean group counts on the five matched nulls per Wikidata variant. Every method is shown before and after the same conditional selection.

Method	W native	W selected	W* native	W* selected
ESCROW recursive	0.0	0.0	0.0	0.0
ESCROW parent	200.0	0.0	204.6	0.0
COBWEB	0.8	0.0	3.8	0.0
FISHDBC	0.8	0.0	1.2	0.0
Agglomerative	0.0	0.0	1.0	0.0
DP-means	0.0	0.0	0.0	0.0
Leader	0.0	0.0	0.0	0.0
BIRCH	16.8	0.0	16.0	0.0
HDBSCAN	2.0	0.0	0.8	0.0
K-means (silhouette)	2.4	0.0	2.6	0.0
GRINCH	458.2	0.0	450.8	0.0
Nested SBM	0.0	0.0	0.0	0.0
Gemini zero-shot	19.2	0.0	17.4	0.0
Gemini few-shot	2.4	0.0	7.8	0.0
Qwen zero-shot	14.6	0.0	10.8	0.0
Qwen few-shot	0.6	0.0	0.0	0.0

Table 12: Complete LLM accounting. Valid means a unique in-chunk response satisfying the contract, including explicit background. Bad JSON counts whole-call parse failures. Limit counts responses stopped at the output allowance. Kept counts streams with at least one selected group.

Model / prompt	Streams	Calls	Valid (%)	Bad JSON	Limit	Signal kept	Null kept
Gemini zero-shot	20	600	99.43	0	0	0/10	0/10
Gemini few-shot	20	600	99.08	5	5	9/10	0/10
Qwen zero-shot	20	600	83.18	28	9	0/10	0/10
Qwen few-shot	20	600	72.12	166	0	0/10	0/10

Every order and control. N and S identify native and conditionally selected graphs. Present loss excludes absent cells. OOV is the percentage of present values mapped to the unknown symbol. It depends on the data and prefix vocabulary, not on the selected graph. The machine-readable outcomes also contain seen-symbol loss and excluded-field counts. Per-order outcomes for every case, arm, selection rule, and prediction head are in `evidence/standard_baselines/derived/matched_prediction/report/rows.json` in the supplementary archive. It holds 6,172 rows at full precision, with the case identifier, corpus, depth, seed, head, selection rule, node count, membership and parent recovery, conditional bits saved, conventional loss, present-cell loss, out-of-vocabulary rate, and excluded-field counts. The paired summaries above aggregate those rows.

D.9 SEQUENTIAL EVIDENCE AND SUPPORTING COMPONENT STUDIES

A separate sequential test rebuilds a two-expert predictor after each block. It scores ten blocks of 100 after 200 warmup records, retaining accumulated evidence across blocks while resetting expert weights. Evidence grows at depths two, three, four, and six; one depth-one seed ends negative. Every null trajectory stays at or below its initial zero. Ten null streams are a diagnostic, not a measurement of a one-in-a-thousand rate.

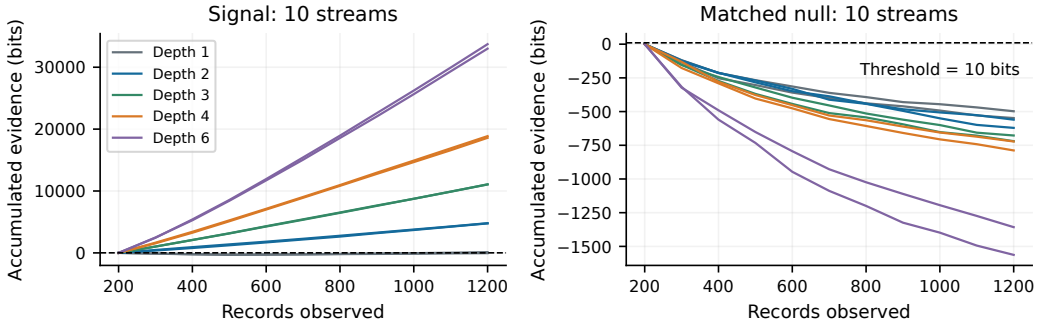


Figure 6: Accumulated block evidence, with initial $W_0 = 1$. Each depth has two signal streams and two matched nulls. Dashed lines mark 10 bits. This component study uses two experts without semantic sharing or forgetting; expert weights reset per block.

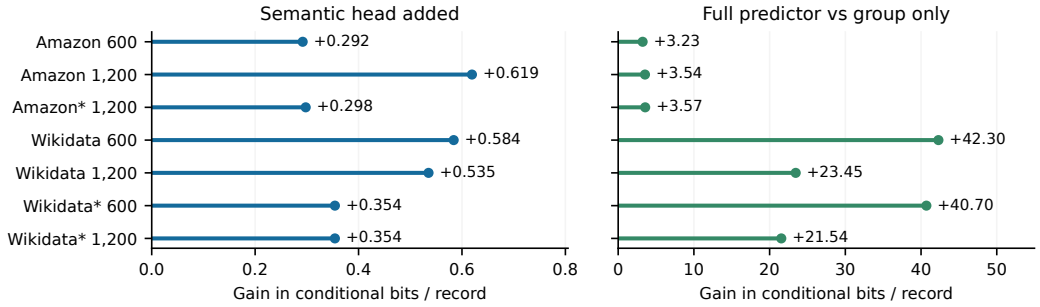


Figure 7: Supporting single-block component studies. The semantic removal holds expert mixing fixed. The group-only removal jointly removes semantic sharing and mixing. The matched factorial table isolates adaptive weighting separately.

D.10 SENSITIVITY TO THE EVIDENCE MARGIN

We replay margins $b \in \{2, 5, 10, 15, 20, 30\}$ on the saved pre-threshold states. Candidate generation and greedy pruning are held fixed. The margin determines whether the resulting graph is retained or replaced by background; it does not tune the search. We retain accepted-graph counts, group counts, recovery, and future predictive scores at every margin.

All 345 pruned decisions in the 115-case primary benchmark are unchanged over this range. At every margin, 123 signal decisions retain groups and no null decision does. Therefore group counts, recovery, and prediction are identical throughout this range for that experiment. This does not establish universal insensitivity. Among the 336 archived whole and pruned decisions, three whole-graph decisions change: one FISHDBC certificate is 29.365 bits, and two parent-insertion certificates are 2.995 and 10.025 bits. None of the archived pruned decisions changes. The complete replay contains 4,086 threshold-specific decisions.

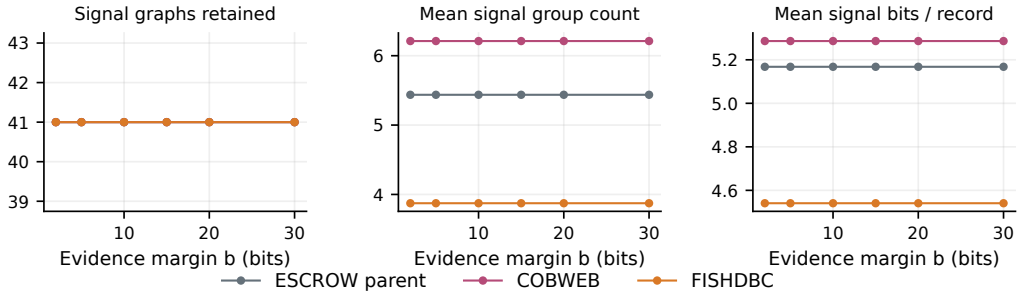


Figure 8: Evidence-margin sensitivity on the 115-case primary benchmark. Each point reapplies the margin to the same pruned graph. All three methods retain their signal decisions and select no groups on every null stream throughout the displayed range.

E SUPPORTING STUDIES AND COMPLETE COMPARISON TABLES

The evidence package retains machine-readable results, including individual seeds where they exist. Tables below report all methods and conditions in each displayed comparison. A missing measurement is shown as a dash, not imputed as zero. Independent-stream replications, algorithm seeds, and prompt variants are kept distinct. Proposal, certification, and readout diagnostics are reported separately in Appendix E.7, E.8, and E.9.

E.1 CLASSICAL NOISE LADDER

Figure 2 summarizes this study; the tables below retain every setting.

The hypothesis is that a useful creation rule should recover groups when repeated cross-field patterns are present and avoid explicit groups when they disappear. The E78 fixture has 3,000 records, eight latent groups, six categorical fields, and three seeds, 0, 1, and 2. Signal controls whether values are drawn from the group's slice of a shared vocabulary or the common noise distribution. Noise is one minus signal. At signal zero, there are no reference groups. Adjusted Rand index (ARI) (Hubert & Arabie, 1985) is therefore omitted at that endpoint; the relevant outcome there is group count.

ESCROW's flat base and split variants use the flat engine. Classical methods receive the same observed records in a normalized multi-hot representation. The supplied source checks the representation and fast distance computations. Rows named true-k receive the known group count. Oracle rows choose the best setting on evaluation labels and are upper references, not deployable competitors. Transferred settings maximize development ARI on a separate full-signal stream, seed 100, and remain fixed across the evaluation ladder. Ties use the first grid value. A method is not described as parameter free merely because its setting was chosen automatically.

DP-means follows Algorithm 1 of Kulis & Jordan (2012). It starts at the global mean, assigns each record to its nearest center, and opens a new center when the squared distance exceeds λ. After each pass it replaces every center by its assigned records' mean, removes empty clusters, and repeats until assignments stabilize. All fits converged before the 1,000-pass limit. Distances within 10^{-12} are numerical ties, resolved by the earliest center; the same tolerance applies to the creation threshold.

BIRCH (Zhang et al., 1996) uses scikit-learn's clustering-feature tree with branching factor 50 and `n_clusters=None`. The threshold bounds the root-mean-square radius after absorbing a record. Tree nodes split when full; 50 is not a cap on the total number of groups. We fit the tree in the declared seeded permutation and assign all records to the final leaf summaries. This configuration does not supply the true group count or apply a final fixed-k clustering step.

Table 13: Parameter grids for the classical ladder. Development selection chooses $\lambda = .4945$ for DP-means and radius .6705 for BIRCH.

Method	Grid
Agglomerative	.05, .1, .2, .3, .4, .42, .44, .45, .46, .47, .48, .49, .495, .5, .7
DP-means	.1, .25, .3, .35, .4, .425, .45, .475, .5, .55, .6, .65, .7, .75, .8, .85, .9, .95, 1, 1.05, 1.1, 1.15, 1.25, 1.5; also $.4800, .4805, \ldots, .5000$.
BIRCH	.2, .35, .4, .45, .5, .55, .6, .625, .65, .675, .7, .725, .75, .8, .825, .85, .88, .92, .95, .98, .99, 1, 1.1, 1.3; also $.6500, .6505, \ldots, .6750$.
Leader	.2, .35, .5, .65, .8, .88, .92, .95, .98, .99, 1, 1.1, 1.3
HDBSCAN	Minimum sizes $5, 25, 100, 200, 400$
K-means	Silhouette-selected $k = 2, \ldots, 20$

The silhouette criterion follows Rousseeuw (1987); an endpoint sensitivity check extends its range to 60. The DP-means and BIRCH settings use only the separate development stream. The complete grids, final assignments, convergence traces, and input hashes are supplied in `evidence/standard_baselines`. The fine grids cover the narrow range of squared distances and subcluster radii in this normalized representation. On the development stream, squared distances from the initial global mean lie between .4879 and .4955. Thus $\lambda \geq .5$ creates no additional centers. The selected $\lambda = .4945$ lies inside that range. An independent dense implementation reproduces the final assignments, centroids, and objectives on all 3,000 development records, including the selected setting.

Table 14: All classical ladder methods: mean ARI at signal 0 through 0.4. Three seeds.

Method	0.00	0.05	0.10	0.20	0.30	0.40
ESCROW flat	n/a	0.000	0.000	0.000	0.000	0.000
ESCROW flat + split	n/a	0.000	0.000	0.000	0.000	0.000
One group per record	n/a	0.000	0.000	0.000	0.000	0.000
One group for all	n/a	0.000	0.000	0.000	0.000	0.000
Key signature	n/a	0.000	0.000	0.000	0.000	0.000
K-means (true k)	n/a	0.000	0.001	0.004	0.006	0.043
Sequential K-means (true k)	n/a	0.000	0.000	0.000	0.000	0.000
Spread K-means (true k)	n/a	0.000	0.000	0.000	0.000	0.000
K-means (silhouette)	n/a	-0.000	0.001	0.003	0.013	0.019
HDBSCAN (default)	n/a	-0.000	-0.000	0.000	-0.000	-0.000
Agglomerative (oracle)	n/a	0.001	0.001	0.005	0.023	0.177
Agglomerative (transfer)	n/a	0.000	0.001	0.003	0.008	0.017
DP-means (oracle)	n/a	0.000	0.001	0.003	0.014	0.053
DP-means (transfer)	n/a	0.000	-0.000	0.000	-0.000	0.000
Leader (oracle)	n/a	0.001	0.001	0.006	0.015	0.029
Leader (transfer)	n/a	0.000	0.001	0.006	0.015	0.029
BIRCH (oracle)	n/a	0.001	0.001	0.004	0.009	0.043
BIRCH (transfer)	n/a	0.000	0.001	0.003	0.006	0.013
HDBSCAN (oracle)	n/a	0.000	0.000	0.000	0.002	0.002
HDBSCAN (transfer)	n/a	-0.000	-0.000	0.000	-0.000	-0.000

Table 15: All classical ladder methods: mean ARI at signal 0.5 through 1.0.

Method	0.50	0.65	0.80	0.90	1.00
ESCROW flat	0.000	0.022	0.377	0.477	0.610
ESCROW flat + split	0.000	0.511	0.744	0.912	0.999
One group per record	0.000	0.000	0.000	0.000	0.000
One group for all	0.000	0.000	0.000	0.000	0.000
Key signature	0.000	0.000	0.000	0.000	0.000
K-means (true k)	0.348	0.859	0.988	0.999	1.000
Sequential K-means (true k)	0.000	0.000	0.002	0.018	0.042
Spread K-means (true k)	0.000	0.000	0.002	0.018	0.042
K-means (silhouette)	0.277	0.755	0.988	0.999	1.000
HDBSCAN (default)	0.008	0.135	0.549	0.866	1.000
Agglomerative (oracle)	0.426	0.746	0.952	0.995	1.000
Agglomerative (transfer)	0.031	0.068	0.150	0.330	1.000
DP-means (oracle)	0.122	0.223	0.382	0.461	0.440
DP-means (transfer)	0.000	0.061	0.211	0.330	0.218
Leader (oracle)	0.052	0.124	0.246	0.367	0.538
Leader (transfer)	0.052	0.124	0.246	0.367	0.538
BIRCH (oracle)	0.137	0.356	0.443	0.563	0.569
BIRCH (transfer)	0.029	0.094	0.260	0.454	0.433
HDBSCAN (oracle)	0.010	0.135	0.549	0.866	1.000
HDBSCAN (transfer)	0.008	0.135	0.549	0.866	1.000

Table 16: Mean returned group counts at signal 0 through 0.4. The pure-noise target is zero.

Method	0.00	0.05	0.10	0.20	0.30	0.40
ESCROW flat	0.000	0.000	0.000	0.000	0.000	0.000
ESCROW flat + split	0.000	0.000	0.000	0.000	0.000	0.000
One group per record	3000.000	3000.000	3000.000	3000.000	3000.000	3000.000
One group for all	1.000	1.000	1.000	1.000	1.000	1.000
Key signature	1.000	1.000	1.000	1.000	1.000	1.000
K-means (true k)	8.000	8.000	8.000	8.000	8.000	8.000
Sequential K-means (true k)	8.000	8.000	8.000	8.000	8.000	8.000
Spread K-means (true k)	8.000	8.000	8.000	8.000	8.000	8.000
K-means (silhouette)	20.000	19.667	20.000	19.667	20.000	20.000
HDBSCAN (default)	2.000	2.000	2.000	2.000	2.000	2.000
Agglomerative (oracle)	1058.333	42.000	180.000	44.333	40.667	20.333
Agglomerative (transfer)	273.333	272.000	274.333	276.667	273.667	262.000
DP-means (oracle)	2126.000	2469.000	2274.667	2043.333	2040.333	1594.000
DP-means (transfer)	18.000	15.333	14.000	15.667	11.667	13.667
Leader (oracle)	3000.000	246.333	42.667	42.333	41.333	42.667
Leader (transfer)	42.667	42.667	42.667	42.333	41.333	42.667
BIRCH (oracle)	1481.667	1540.000	1434.667	1133.333	777.000	487.000
BIRCH (transfer)	1399.333	1349.000	1355.000	1425.333	1421.667	1349.333
HDBSCAN (oracle)	0.667	2.000	2.000	2.000	2.000	2.000
HDBSCAN (transfer)	2.000	2.000	2.000	2.000	2.000	2.000

Table 17: Mean returned group counts at signal 0.5 through 1.0. The planted count is eight.

Method	0.50	0.65	0.80	0.90	1.00
ESCROW flat	0.333	0.333	2.667	3.667	5.000
ESCROW flat + split	0.333	5.000	6.000	6.667	7.667
One group per record	3000.000	3000.000	3000.000	3000.000	3000.000
One group for all	1.000	1.000	1.000	1.000	1.000
Key signature	1.000	1.000	1.000	1.000	1.000
K-means (true k)	8.000	8.000	8.000	8.000	8.000
Sequential K-means (true k)	8.000	8.000	8.000	8.000	8.000
Spread K-means (true k)	8.000	8.000	8.000	8.000	8.000
K-means (silhouette)	19.000	14.000	8.000	8.000	8.000
HDBSCAN (default)	28.000	68.667	17.333	9.333	8.000
Agglomerative (oracle)	8.667	8.000	9.667	8.333	8.000
Agglomerative (transfer)	239.333	185.000	104.000	44.000	8.000
DP-means (oracle)	2107.667	1841.000	1261.667	788.000	580.000
DP-means (transfer)	22.000	468.000	729.667	475.000	261.667
Leader (oracle)	43.333	42.667	43.000	41.667	45.333
Leader (transfer)	43.333	42.667	43.000	41.667	45.333
BIRCH (oracle)	407.667	248.667	279.333	227.000	119.667
BIRCH (transfer)	1193.333	1006.333	603.667	281.000	154.333
HDBSCAN (oracle)	26.000	68.667	17.333	9.333	8.000
HDBSCAN (transfer)	28.000	68.667	17.333	9.333	8.000

These results support a tradeoff between recovery and abstention. They do not establish that every baseline hallucinates groups. HDBSCAN settings can return no groups; the cost is visible on signal. The separate shortest-of-several-order ESCROW experiment also creates null groups, so the single-order result cannot be transferred to that selection policy without paying for the additional choice.

E.2 Matched LLM noise ladder

E82 uses 300 records per rung, seed 0, opaque names, and six additional irrelevant fields. Both methods can leave records ungrouped. The hypothesis is the same as above, but the input size, available signal, and implementation differ. Every rung returns zero groups for both ESCROW and Gemini. This is not an ESCROW recovery advantage. Empty predictions score zero membership F1 when signal groups exist; null F1 is omitted because the empty/empty value is only a convention.

Table 18: Matched small-stream noise ladder. Both methods abstain at every rung.

Signal	ESCROW groups	F1	Gemini groups	F1
0	0	n/a	0	n/a
0.05	0	0.000	0	0.000
0.1	0	0.000	0	0.000
0.2	0	0.000	0	0.000
0.3	0	0.000	0	0.000
0.4	0	0.000	0	0.000
0.5	0	0.000	0	0.000
0.65	0	0.000	0	0.000
0.8	0	0.000	0	0.000
0.9	0	0.000	0	0.000
1	0	0.000	0	0.000

E.3 Depth recovery and its controls

Figure 3 compares native and conditionally pruned graphs on matched inputs. Appendix E.3.1 specifies that complete comparison. Figure 10 separately retains the EHR05 baseline filters. The EHR05 code rule preserves every native FISHDBC membership and link in the 18 signal cases at depths 2–7. This explains their coincident curves; it is not rounding. Their null outputs differ: native

FISHDBC returns groups on all 21 nulls, while its coded arm returns none. The EHR05 rule is separate from the conditional certifier used for the insertion-based prediction pipeline.

Table 19: Recursive cascade confirmation on 20 reserved streams per row. Pred. / Ref. is mean node count, not the number of correctly recovered groups. Larger samples improve recovery but leave many reference groups and links missing.

Depth	n	Member F1	Parent F1	Anc. F1	Height	Pred. / Ref.
2	300	0.999	1.000	1.000	2 to 2	6 / 6
4	300	0.807	0.473	0.376	2 to 3	12.25 / 30
7	300	0.539	0.051	0.022	2 to 3	12.25 / 242.1
7	1,000	0.690	0.183	0.140	3 to 5	45.15 / 253.95

The development fixture has three informative categorical fields per level and six irrelevant fields. Seeds 34000, 34001, and 34002 generate depths one through seven. Each signal stream has a paired independently column-shuffled null. All compared methods receive the same 300 records in the same order. Repairs are available every 100 records and once at the end. The ESCROW arm is the recursive cascade; the real-data experiment uses parent-based insertion. Appendix E.9 compares them on these same inputs.

FISHDBC uses a distance computed from the observed categorical records, with `min_samples=5`, `m=5`, `ef=50`, and a final minimum cluster size of five. The native arm exports its hierarchy. The coded arm applies the archived EHR05 whole-candidate replacement test. It prices the full proposed hierarchy and keeps it only when both its description and serialized form improve on the prior state. It does not use the EHR07 conditional 10-bit certificate or its greedy pruning routine. Theorem 1 must not be retroactively assigned to this arm. All input hashes and protocol settings are retained in the campaign records.

Reference and predicted nodes are paired by maximum summed membership F1 under one-to-one matching. Membership precision and recall are then computed over (group, record) pairs under that correspondence; unmatched memberships remain errors. A high membership F1 does not mean that the same fraction of reference nodes was recovered. Direct-parent and ancestor F1 use the same correspondence. Height is the longest returned path and can be too large even when membership is poor.

The raw evaluator assigns empty/empty F1 the value one. At depth one there are no reference links, so the displayed parent and ancestor F1 are n/a for every method. Table 21 reports all predicted links in that condition as false links instead. Null membership F1 is also omitted; null group count is the relevant error outcome. These are reporting conventions, with the raw metrics retained.

Table 20: Matched development results, means over three signal streams per depth. The frozen file retains every signal and null stream. Relation F1 is n/a at depth one, where the reference has no links.

Depth	Method	Group F1	Parent F1	Ancestor F1	Height
1	ESCROW cascade	0.805	n/a	n/a	1.67
1	FISHDBC	0.679	n/a	n/a	3.67
1	FISHDBC + code	0.234	n/a	n/a	1.33
1	Gemini	0.000	n/a	n/a	0.00
2	ESCROW cascade	1.000	1.000	1.000	2.00
2	FISHDBC	0.738	0.204	0.179	4.67
2	FISHDBC + code	0.738	0.204	0.179	4.67
2	Gemini	0.444	0.000	0.000	0.67
3	ESCROW cascade	0.877	0.700	0.611	2.67
3	FISHDBC	0.867	0.839	0.752	4.00
3	FISHDBC + code	0.867	0.839	0.752	4.00
3	Gemini	0.351	0.095	0.061	1.00
4	ESCROW cascade	0.788	0.390	0.326	3.00
4	FISHDBC	0.836	0.686	0.666	4.00
4	FISHDBC + code	0.836	0.686	0.666	4.00
4	Gemini	0.267	0.000	0.000	0.67
5	ESCROW cascade	0.706	0.261	0.156	3.00
5	FISHDBC	0.795	0.449	0.423	4.67
5	FISHDBC + code	0.795	0.449	0.423	4.67
5	Gemini	0.282	0.000	0.000	1.00
6	ESCROW cascade	0.571	0.070	0.046	2.67
6	FISHDBC	0.736	0.267	0.248	5.00
6	FISHDBC + code	0.736	0.267	0.248	5.00
6	Gemini	0.190	0.000	0.000	0.67
7	ESCROW cascade	0.553	0.067	0.026	3.00
7	FISHDBC	0.676	0.167	0.119	5.00
7	FISHDBC + code	0.676	0.167	0.119	5.00
7	Gemini	0.184	0.006	0.001	1.00

Table 21: Depth-one signal streams, whose reference has no hierarchy links. All returned parent and ancestor links are false in this diagnostic. Zero links do not establish successful group recovery. Counts are recomputed from the saved graphs.

Method	Seed	Nodes	False parents	False ancestors
ESCROW cascade	34000	4	2	2
FISHDBC	34000	12	10	18
FISHDBC + code	34000	0	0	0
Gemini	34000	0	0	0
ESCROW cascade	34001	3	0	0
FISHDBC	34001	8	6	12
FISHDBC + code	34001	8	6	12
Gemini	34001	0	0	0
ESCROW cascade	34002	4	2	2
FISHDBC	34002	10	8	12
FISHDBC + code	34002	0	0	0
Gemini	34002	0	0	0

Native FISHDBC returns 6 to 20 groups on each null endpoint; the filter returns zero on all 21. It also returns zero at all 84 recorded null readouts, which are four repeated observations per stream, not 84 independent streams. At signal endpoints, it retains memberships and parent links in 19 of 21 cases and rejects two depth-one candidates. Field supports and ownership can change during pricing, so we do not call these 19 complete graph objects identical. The result supports a useful whole-graph filter. Appendix E.7 separately tests pruning with the conditional code on these native proposals; it does not retroactively change the archived filter.

A separate fixed-eight-leaf diagnostic found a supplied seven-level graph cheaper than the depth-four graph returned by search. This exposes a search limit, while other fine distinctions can lack enough observations to pay their cost. Neither explains every failure.

The complete Gemini development campaign comprises 42 completed streams collected across the original run, continuation, and final transport recovery. Each stream has 300 arrival calls and a terminal repair. Invalid replies, transport failures, and resumed work remain in the saved logs. The original per-call output cap is 16,384 tokens; a separate terminal control raises it to 65,536 without adding a repair or regenerating prior observations. That paired control tests response capacity, not a new independent stream. The model still produces at most two levels. Neither arm is allowed access to field roles, true depth, groups, or labels.

A preceding context-management probe changes how history is presented on the same streams. Compact context returns depth two and membership F1 .753 on the depth-two case, versus an empty output under repeated history. At depth four it returns no groups, versus depth one and F1 .308 under the one-record context protocol. On the field-disclosed depth-two control both return depth one and F1 .667. Both return empty on the independent null. These mixed outcomes motivate reporting context management rather than attributing all failures to model capacity.

The cascade confirmation uses 20 reserved streams per reported condition, at depths two, four, and seven. At depth seven the 300- and 1,000-record results are prefixes of the same streams. Table 19 gives the four aggregate rows. The full 80 signal rows, 20 depth-seven null rows, and per-level matches are in `depth_confirmation.json`. This arm corrected a feasibility constraint after the original confirmation campaign began. Its amended protocol states that only completion counts, not confirmation recovery metrics, had been inspected, and freezes the correction before its outcomes. It is an amended confirmation, not an unchanged preregistered campaign. With zero errors in 20 independent null streams, a one-sided exact 95% binomial upper limit is $1 - .05^{1/20} = .139$. Such a small empirical audit cannot establish a .001 error rate. The theorem, under its own assumptions, is the source of a small error bound.

Matched COBWEB comparison. COBWEB is evaluated on the same 42 archived inputs and record order as the other methods in Figure 3: depths 1–7, seeds 34000–34002, 300 records, signal and paired column-shuffled nulls. This completes the baseline set on these development inputs; it is not an independent confirmation sample. The protocol was frozen before executing the additional baseline. All outcomes are retained.

We use `concept-formation 0.3.4` with default `CobwebTree`, one `ifit` call per arriving record, and the input seed for tie breaking. At prefixes 100, 200, and 300, plus a terminal readout, `categorize` maps every observed record to a leaf. Its ancestors supply group memberships. Readout preserves the fitting random state. Export keeps every distinct nonuniversal group with at least two records, with parent links from minimal strict inclusion, exactly as in the matched prediction adapter. FISHDBC retains its declared minimum cluster size five. No evaluation label selects a cut or group count for either baseline.

The COBWEB + code arm applies the *same EHR05 whole-candidate replacement rule* as FISHDBC + code at each checkpoint. Supports and ownership use the same adapter. The candidate replaces the retained graph only if both its complete ideal description and its serialized byte length decrease. This is distinct from conditional pruning in the prediction study. The native graph is always reported alongside it. At depths 1–3 the coded graph stays empty; at depths 4–7 its memberships and links equal the native graph's. For example, at depth two, seed 34000, the 122-group candidate costs 2,168.2 extra ideal bits and 2,168 extra serialized bits at 300 records. This rejection follows the whole-graph rule; it does not establish that every COBWEB group lacks support.

All 168 checkpoint decisions and serialized descriptions are independently reconstructed. The final native graphs at depths two and seven, seed 34000, also match replays through the existing prediction adapter. Membership and direct-link counts are recomputed at the saved matching. The complete source, graph states, code terms, and protocol hashes accompany the following terminal results.

Table 22: Every matched COBWEB signal case. N: returned groups; H: returned depth. F1 is membership F1; Parent and Anc. are direct-link and ancestor F1. Relationship F1 is omitted at depth one, where the reference has no links.

		COBWEB native					COBWEB + code				
Depth	Seed	N	F1	Parent	Anc.	H	N	F1	Parent	Anc.	H
1	34000	115	0.4280	n/a	n/a	7	0	0.0000	n/a	n/a	0
1	34001	112	0.4167	n/a	n/a	6	0	0.0000	n/a	n/a	0
1	34002	125	0.3992	n/a	n/a	5	0	0.0000	n/a	n/a	0
2	34000	122	0.6218	0.0645	0.0199	6	0	0.0000	0.0000	0.0000	0
2	34001	125	0.6237	0.0630	0.0196	6	0	0.0000	0.0000	0.0000	0
2	34002	127	0.6214	0.0620	0.0190	6	0	0.0000	0.0000	0.0000	0
3	34000	131	0.7268	0.1702	0.0881	6	0	0.0000	0.0000	0.0000	0
3	34001	129	0.7571	0.1727	0.0883	6	0	0.0000	0.0000	0.0000	0
3	34002	139	0.7397	0.1611	0.0805	6	0	0.0000	0.0000	0.0000	0
4	34000	135	0.8074	0.3478	0.2455	7	135	0.8074	0.3478	0.2455	7
4	34001	127	0.7094	0.2895	0.2258	6	127	0.7094	0.2895	0.2258	6
4	34002	135	0.7679	0.2981	0.2392	6	135	0.7679	0.2981	0.2392	6
5	34000	132	0.7567	0.3895	0.4942	7	132	0.7567	0.3895	0.4942	7
5	34001	127	0.7604	0.3676	0.5165	6	127	0.7604	0.3676	0.5165	6
5	34002	131	0.7867	0.3810	0.5116	6	131	0.7867	0.3810	0.5116	6
6	34000	135	0.7406	0.3059	0.6508	6	135	0.7406	0.3059	0.6508	6
6	34001	139	0.7459	0.3678	0.6581	7	139	0.7459	0.3678	0.6581	7
6	34002	130	0.7252	0.3520	0.6519	6	130	0.7252	0.3520	0.6519	6
7	34000	127	0.7194	0.2534	0.4684	6	127	0.7194	0.2534	0.4684	6
7	34001	124	0.6839	0.1939	0.4220	5	124	0.6839	0.1939	0.4220	5
7	34002	132	0.6554	0.2259	0.4331	7	132	0.6554	0.2259	0.4331	7

Table 23: Every matched COBWEB null case. Both native and filtered outputs are shown. All 21 coded outputs are empty. Raw native groups are not support-certified.

Depth	Seed	Native N	Native H	Coded N	Coded H
1	34000	122	5	0	0
1	34001	115	5	0	0
1	34002	120	5	0	0
2	34000	119	5	0	0
2	34001	133	6	0	0
2	34002	127	5	0	0
3	34000	116	4	0	0
3	34001	118	5	0	0
3	34002	120	5	0	0
4	34000	110	5	0	0
4	34001	104	4	0	0
4	34002	111	4	0	0
5	34000	115	5	0	0
5	34001	100	5	0	0
5	34002	102	4	0	0
6	34000	105	4	0	0
6	34001	100	5	0	0
6	34002	96	5	0	0
7	34000	115	5	0	0
7	34001	108	5	0	0
7	34002	88	5	0	0

E.3.1 THE SAME CONDITIONAL SELECTION ACROSS ALL FOUR PROPOSERS

Question and fixed protocol. Does a recovery difference come from the returned candidate graph or from its selection rule? We compare recursive ESCROW, parent-based insertion, COBWEB, and FISHDBC on the identical 42 depth inputs above. Each contributes three arms: its native returned graph, whole conditional certification, and conditional certification with pruning. This gives 504 arm outcomes, with 21 signal and 21 null streams for every arm. The native recursive graph has already passed that proposer's own EHR05 search gate. Native does not mean an unfiltered record of every candidate it considered.

Every conditional arm uses `FastCertifier(prefix, values="min")`, the same support search, a 10-bit margin, and explicit selected nodes as its readout. Whole and pruned arms start from the same memberships and parent lists. The whole arm calls `certify(prune=False)`; the pruned arm changes only this option to `True`. Neither is the EHR05 replacement rule. The certificate is tested once at the 300-record endpoint, without a repeated-prefix claim. No reference labels enter support choice, pruning, or acceptance.

The FISHDBC and two ESCROW comparisons reuse their saved conditional outcomes. Their input records, candidate graphs, and certifier source hashes match the declared inputs and unchanged code. The fixed-pool comparison applies both conditional arms to each saved terminal COBWEB graph. Its protocol and all input, graph, and source hashes were frozen before execution. No proposer or LLM was rerun. All 42 cases completed. This analysis reuses development inputs; it is not an independent confirmation sample, and no setting was selected from the recovery results.

What selection recovers. At depth two, native COBWEB supplies 122, 125, and 127 groups. Whole conditional selection rejects all three pools. Pruning removes 116, 119, and 121 groups, respectively, and retains exactly the six reference memberships and every parent and ancestor link. Mean membership F1 changes from 0.6223 to 1.0000. At depth one, pruned COBWEB is exact on two streams and retains four groups on the third; its mean is 0.9362. Pruned FISHDBC is exact on all three depth-one streams. All four proposers' whole and pruned conditional arms return no groups on all 21 nulls.

This comparison identifies a useful contribution of optional pruning within the conditional code: a large rejected pool can contain a supported, accurate subset. It does not attribute recovery to certification alone, since candidate groups are still supplied by search. Nor does a global-null certificate prove each selected group correct. The EHR05-to-conditional change alters the coding rule; the paired conditional whole-versus-pruned arms isolate the pruning option.

How the ranking changes. Native recursive ESCROW has the highest mean membership F1 at depths one through three among the native arms. That statement does not extend to the conditionally pruned comparison. There, FISHDBC leads at depth one, COBWEB at depths two and three, recursive ESCROW at depth four, and FISHDBC at depths five through seven. The depth-two difference between COBWEB and recursive ESCROW is only 0.00028; these three-stream means are descriptive rankings, not evidence of population superiority. Recursive recovery remains much stronger than parent-based insertion on the matched inputs. No predictive ranking between those two proposers follows from this recovery study.

Table 24: Identical streams, four proposers, and three selection states. F1 denotes membership recovery. Parent, ancestor, group count N, and depth H refer to the pruned arm. The native and whole arms' complete metrics remain in the source data and their individual study tables. Means over three signal streams per depth.

Depth	Proposer	Native F1	Whole F1	Pruned F1	Parent	Anc.	N	H
1	ESCROW recursive	0.8046	0.8046	0.8046	n/a	n/a	3.67	1.67
1	ESCROW parent	0.2792	0.0000	0.0000	n/a	n/a	0.00	0.00
1	COBWEB	0.4146	0.0000	0.9362	n/a	n/a	2.67	1.33
1	FISHDBC	0.6788	0.2342	1.0000	n/a	n/a	2.00	1.00
2	ESCROW recursive	0.9997	0.9997	0.9997	1.0000	1.0000	6.00	2.00
2	ESCROW parent	0.4856	0.0000	0.6122	0.0000	0.0000	5.00	1.00
2	COBWEB	0.6223	0.0000	1.0000	1.0000	1.0000	6.00	2.00
2	FISHDBC	0.7380	0.7380	0.8620	0.7302	0.7302	4.33	2.00
3	ESCROW recursive	0.8773	0.8773	0.8667	0.6789	0.5833	8.33	2.67
3	ESCROW parent	0.4453	0.1484	0.4795	0.0000	0.0000	7.67	1.00
3	COBWEB	0.7412	0.0000	0.8939	0.7485	0.6784	9.33	3.00
3	FISHDBC	0.8666	0.8666	0.8000	0.5000	0.3333	6.00	2.00
4	ESCROW recursive	0.7884	0.7884	0.7884	0.3899	0.3263	11.33	3.00
4	ESCROW parent	0.4039	0.4039	0.3740	0.0000	0.0000	11.00	1.00
4	COBWEB	0.7616	0.7616	0.7844	0.4665	0.3648	12.67	3.00
4	FISHDBC	0.8363	0.8363	0.7553	0.4131	0.2695	9.33	3.00
5	ESCROW recursive	0.7058	0.7058	0.7058	0.2614	0.1558	12.67	3.00
5	ESCROW parent	0.4079	0.4079	0.3191	0.0000	0.0000	16.33	1.00
5	COBWEB	0.7679	0.7679	0.6843	0.2254	0.2043	15.67	4.00
5	FISHDBC	0.7954	0.7954	0.7225	0.3009	0.1648	12.67	3.33
6	ESCROW recursive	0.5712	0.5712	0.5659	0.0703	0.0445	11.00	2.67
6	ESCROW parent	0.3896	0.3896	0.2748	0.0000	0.0000	19.00	1.00
6	COBWEB	0.7372	0.7372	0.6142	0.1251	0.1309	23.33	4.00
6	FISHDBC	0.7359	0.7359	0.6473	0.1669	0.0833	14.33	3.67
7	ESCROW recursive	0.5526	0.5526	0.5526	0.0670	0.0258	12.33	3.00
7	ESCROW parent	0.3393	0.3393	0.2383	0.0000	0.0006	22.00	1.33
7	COBWEB	0.6862	0.6862	0.5542	0.0744	0.0603	25.00	3.33
7	FISHDBC	0.6761	0.6761	0.5862	0.0880	0.0361	14.33	4.00

Deeper recovery. Pruning is not uniformly beneficial. COBWEB improves mean membership recovery through depth four, then loses recovery at depths five through seven. FISHDBC improves at depths one and two and loses recovery at depths three through seven. Across the 21 COBWEB streams, 11 improve and 10 worsen; median gain is 0.0218, compared with mean gain 0.1052. FISHDBC and parent-based insertion each improve on six streams and worsen on fifteen. Their mean changes are 0.0066 and -0.0647. These cross-depth means summarize the declared conditions, not a uniform improvement. Recursive pruning changes only two signal outputs slightly; whole conditional certification preserves every native recursive signal graph. The conditional criterion selects a supported description, which need not maximize recovery of every planted group or relationship. Figure 9 reports returned depth using the same colors and methods as the main comparison.

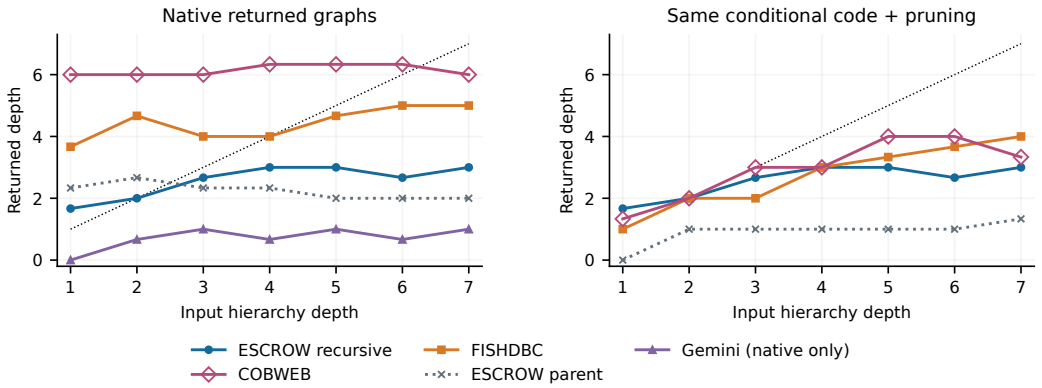


Figure 9: Returned depth before and after the same conditional pruning. The dotted diagonal is input depth. A high returned depth alone does not establish accurate membership or relationships. Gemini is shown only for its measured native output; no conditional Gemini result is inferred.

Verification and matching sensitivity. The audit reconstructs all 336 conditional certificates using both the vectorized and scalar model-cost implementations. Their largest disagreement is 7.3×10^{-12} bits. It checks the common installed pools, threshold decisions, positive deletion savings, selected graphs, and all 504 recovery outcomes. The depth-two COBWEB memberships and links are also compared directly with the supplied reference sets after selection. No matching tie affects the COBWEB conditional arms. Independent direct-F1 arithmetic finds equally optimal assignments with different reported scores in six reused native/whole arms: FISHDBC at depth five, seed 34000, and insertion at depth six, seeds 34001 and 34002. Primary and alternative scores are retained. They do not change the stated shallow conclusions or mean native ranking. We preserve the original declared matching throughout.

Table 25: Every COBWEB signal stream under whole conditional selection. N and H describe the selected graph. Certificate is the score of the candidate before thresholding; a rejected pool therefore has a negative score even though its selected graph is empty. Relationship F1 is omitted at depth one.

Depth	Seed	N	Member F1	Parent F1	Anc. F1	H	Certificate
1	34000	0	0.0000	n/a	n/a	0	-2728.61
1	34001	0	0.0000	n/a	n/a	0	-2588.98
1	34002	0	0.0000	n/a	n/a	0	-2915.54
2	34000	0	0.0000	0.0000	0.0000	0	-1828.02
2	34001	0	0.0000	0.0000	0.0000	0	-1931.87
2	34002	0	0.0000	0.0000	0.0000	0	-1978.84
3	34000	0	0.0000	0.0000	0.0000	0	-579.32
3	34001	0	0.0000	0.0000	0.0000	0	-228.16
3	34002	0	0.0000	0.0000	0.0000	0	-631.24
4	34000	135	0.8074	0.3478	0.2455	7	1234.72
4	34001	127	0.7094	0.2895	0.2258	6	1279.05
4	34002	135	0.7679	0.2981	0.2392	6	1227.91
5	34000	132	0.7567	0.3895	0.4942	7	2230.06
5	34001	127	0.7604	0.3676	0.5165	6	2448.28
5	34002	131	0.7867	0.3810	0.5116	6	2600.74
6	34000	135	0.7406	0.3059	0.6508	6	3143.54
6	34001	139	0.7459	0.3678	0.6581	7	2487.04
6	34002	130	0.7252	0.3520	0.6519	6	3047.72
7	34000	127	0.7194	0.2534	0.4684	6	3055.88
7	34001	124	0.6839	0.1939	0.4220	5	3083.35
7	34002	132	0.6554	0.2259	0.4331	7	2724.14

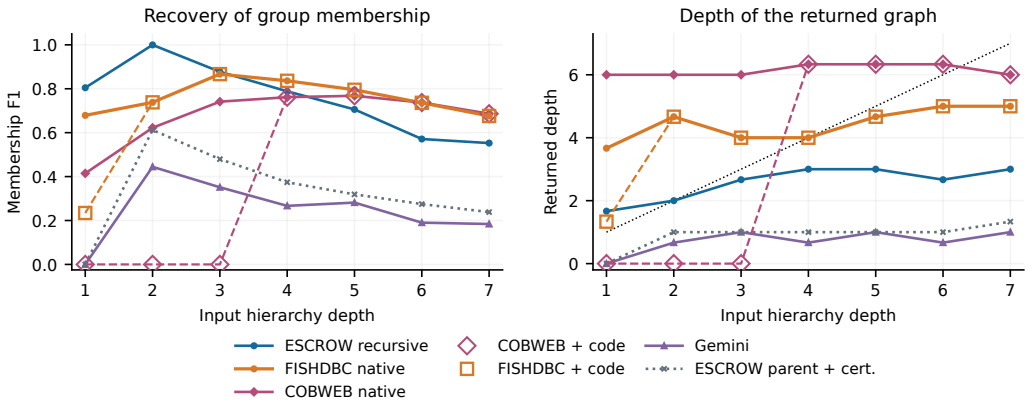
Table 26: Every COBWEB signal stream under the same conditional code with pruning. No candidate membership is changed or created; selection removes nodes and updates supports and routing. All retained parent and ancestor metrics are shown.

Depth	Seed	N	Member F1	Parent F1	Anc. F1	H	Certificate
1	34000	2	1.0000	n/a	n/a	1	407.80
1	34001	2	1.0000	n/a	n/a	1	407.95
1	34002	4	0.8086	n/a	n/a	2	342.07
2	34000	6	1.0000	1.0000	1.0000	2	1499.93
2	34001	6	1.0000	1.0000	1.0000	2	1497.72
2	34002	6	1.0000	1.0000	1.0000	2	1479.99
3	34000	7	0.8258	0.5882	0.4615	3	2757.16
3	34001	11	0.9349	0.8571	0.8235	3	3030.37
3	34002	10	0.9210	0.8000	0.7500	3	2893.31
4	34000	14	0.8571	0.6000	0.4545	3	4618.83
4	34001	12	0.7062	0.3784	0.2785	3	4488.66
4	34002	12	0.7897	0.4211	0.3614	3	4640.46
5	34000	16	0.6690	0.2162	0.2100	4	5248.86
5	34001	14	0.6720	0.1667	0.1767	4	5546.31
5	34002	17	0.7119	0.2933	0.2262	4	5639.22
6	34000	23	0.6220	0.1678	0.1179	4	6017.84
6	34001	25	0.6250	0.1088	0.1569	4	5423.86
6	34002	22	0.5954	0.0986	0.1179	4	5867.28
7	34000	24	0.5852	0.1000	0.0657	4	5535.74
7	34001	26	0.5541	0.0608	0.0608	3	5467.28
7	34002	25	0.5233	0.0625	0.0545	3	5108.50

Table 27: Every matched COBWEB null. Both conditional arms select zero groups. Certificates describe the whole or pruned candidate before thresholding.

Depth	Seed	Native N	Whole N	Pruned N	Whole cert.	Pruned cert.
1	34000	122	0	0	-3422.01	-330.40
1	34001	115	0	0	-3257.91	-255.91
1	34002	120	0	0	-3373.31	-423.99
2	34000	119	0	0	-3488.18	-242.14
2	34001	133	0	0	-3870.21	-365.15
2	34002	127	0	0	-3638.10	-261.94
3	34000	116	0	0	-3638.45	-446.13
3	34001	118	0	0	-3579.37	-142.09
3	34002	120	0	0	-3848.16	-1.00
4	34000	110	0	0	-3624.17	-1.00
4	34001	104	0	0	-3371.25	-405.64
4	34002	111	0	0	-3593.34	-289.50
5	34000	115	0	0	-3820.57	-351.63
5	34001	100	0	0	-3510.87	-1.00
5	34002	102	0	0	-3361.49	-362.33
6	34000	105	0	0	-3536.67	-315.59
6	34001	100	0	0	-3377.55	-325.95
6	34002	96	0	0	-3293.83	-420.90
7	34000	115	0	0	-3918.37	-1.00
7	34001	108	0	0	-3731.70	-427.01
7	34002	88	0	0	-3109.29	-342.21

Complete-code replacement as a separate comparison. Figure 10 retains the native and EHR05-filtered baseline results. They answer a different selection question from the conditional pruning in Figure 3. Open markers distinguish filtered outcomes from native outcomes where their trajectories coincide; color identifies the proposer.



Groups on matched nulls	Input depth:	1	2	3	4	5	6	7
ESCROW recursive		0	0	0	0	0	0	0
FISHDBC native		18.0	14.7	12.7	9.3	11.3	8.0	9.3
COBWEB native		119.0	126.3	118.0	108.3	105.7	100.3	103.7
COBWEB + code		0	0	0	0	0	0	0
FISHDBC + code		0	0	0	0	0	0	0
Gemini		0	0	0	0	0	0	0
ESCROW parent + cert.		0	0	0	0	0	0	0

Figure 10: Native and EHR05-filtered baseline graphs on the same inputs. The insertion arm uses conditional pruning, as labelled; it is not an EHR05-filtered arm. The baseline EHR05 replacement rule retains FISHDBC signal memberships and links at depths 2–7 and COBWEB's at depths 4–7, explaining the overlapping markers.

E.4 Real-data prediction and LLM prompting

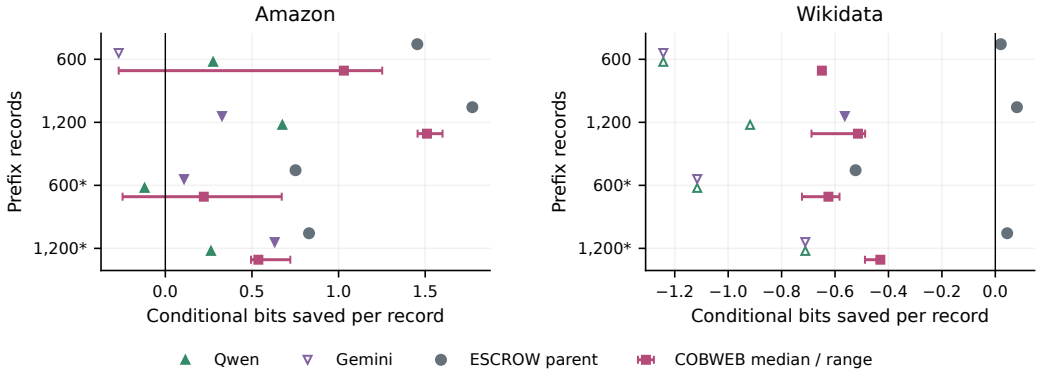


Figure 11: Single-block real-data comparison: one evaluation block per configuration. COBWEB marks are medians and observed five-seed ranges. Hollow LLM marks denote empty selected graphs. The matched multiple-order, three-block study is reported separately.

The hypothesis is that a useful proposed graph improves the description of new records when downstream evaluation is held fixed. The two corpora are Amazon Berkeley Objects product records and Wikidata people records. The stored processed record order is retained. Field and value objects are converted to strings by the shared loader. The corresponding curated paths produce reference groups from every nontrivial path prefix. They are comparison structures, not labels supplied to the unsupervised proposers.

The field-removal versions use the repository's `abo/without_product_type` and `wikidata_hier/without_partial_leakers` data directories. They remove fields that expose parts of the curated classification. These variants share records and should not be treated as

independent datasets. Prefixes contain 600 or 1,200 records; the next 100 records form one evaluation block.

Each proposer has the same prefix, certifier, evidence margin, shared-count predictor, and evaluation block. COBWEB has five algorithm seeds, 0 through 4. The three ESCROW repeats give the same score in these cases; they are not three independent data draws. LLM scores are individual runs. The reported five-seed range describes COBWEB's observed algorithm variation and is not a confidence interval for performance on unseen streams.

Table 28: Conditional bits saved per evaluation record. Asterisks denote field removal. COBWEB is its five-seed mean, including empty outcomes. Amazon 600 and Amazon* 600 each include one empty seed among five; the other means include none. † marks an empty selected LLM graph. ‡ recovers the Amazon* 600 empty score from COBWEB seed 4, which selected zero nodes under the same scorer; it is not a separately rerun control. The curated score for that configuration remains missing.

Corpus	n	Empty	Curated	Gemini	Qwen	COBWEB mean	ESCROW
Amazon	600	-0.269	0.536	$-0.269^{\dagger}$	0.276	0.777	1.454
Amazon	1,200	-0.256	1.109	0.327	0.675	1.517	1.772
Amazon*	600	$-0.247^{\ddagger}$	–	0.108	-0.120	0.204	0.752
Amazon*	1,200	-0.246	0.161	0.631	0.263	0.589	0.829
Wikidata	600	-1.243	-0.803	$-1.243^{\dagger}$	$-1.243^{\dagger}$	-0.650	0.020
Wikidata	1,200	-0.918	-0.274	-0.563	$-0.918^{\dagger}$	-0.557	0.081
Wikidata*	600	-1.116	-1.051	$-1.116^{\dagger}$	$-1.116^{\dagger}$	-0.643	-0.523
Wikidata*	1,200	-0.711	-0.560	$-0.711^{\dagger}$	$-0.711^{\dagger}$	-0.448	0.044

An empty graph still defines a block predictor with background counts and the declared mixture. Its score need not be zero, since zero refers to the untilted conditional reference. The Amazon* 600 recovered score is -0.2466872763 bits per record. For all other observed empty COBWEB outcomes, the score equals the separately saved empty control. We retain failed seeds when averaging over the declared algorithm.

Table 29: COBWEB variation across five seeds. Nonempty counts seeds retaining at least one certified group. ESCROW repeats have identical selected node counts and scores.

Corpus	n	Mean	Median	Min	Max	Nonempty	ESCROW nodes
Amazon	600	0.777	1.030	-0.269	1.252	4/5	11
Amazon	1,200	1.517	1.510	1.456	1.600	5/5	17
Amazon*	600	0.204	0.222	-0.247	0.672	4/5	11
Amazon*	1,200	0.589	0.537	0.495	0.721	5/5	15
Wikidata	600	-0.650	-0.649	-0.654	-0.649	5/5	14
Wikidata	1,200	-0.557	-0.514	-0.688	-0.487	5/5	30
Wikidata*	600	-0.643	-0.625	-0.724	-0.583	5/5	11
Wikidata*	1,200	-0.448	-0.430	-0.488	-0.422	5/5	25

Table 30: Every Amazon COBWEB seed. Proposed counts precede certification; certified counts describe the selected graph. Scores are conditional bits saved per record.

Corpus	n	Seed	Proposed	Certified	Bits / rec.
Amazon	600	0	322	11	0.791
Amazon	600	1	315	16	1.252
Amazon	600	2	315	14	1.081
Amazon	600	3	313	0	-0.269
Amazon	600	4	330	17	1.030
Amazon	1,200	0	613	37	1.456
Amazon	1,200	1	639	31	1.510
Amazon	1,200	2	634	32	1.529
Amazon	1,200	3	628	33	1.600
Amazon	1,200	4	627	33	1.491
Amazon*	600	0	305	5	0.026
Amazon*	600	1	316	10	0.344
Amazon*	600	2	304	10	0.222
Amazon*	600	3	306	14	0.672
Amazon*	600	4	307	0	-0.247
Amazon*	1,200	0	584	12	0.506
Amazon*	1,200	1	587	22	0.537
Amazon*	1,200	2	596	32	0.495
Amazon*	1,200	3	587	24	0.684
Amazon*	1,200	4	596	34	0.721

Table 31: Every Wikidata COBWEB seed under the same reporting convention.

Corpus	n	Seed	Proposed	Certified	Bits / rec.
Wikidata	600	0	114	6	-0.649
Wikidata	600	1	113	6	-0.649
Wikidata	600	2	113	6	-0.649
Wikidata	600	3	113	6	-0.649
Wikidata	600	4	111	6	-0.654
Wikidata	1,200	0	275	17	-0.688
Wikidata	1,200	1	244	12	-0.487
Wikidata	1,200	2	247	12	-0.487
Wikidata	1,200	3	252	15	-0.514
Wikidata	1,200	4	242	17	-0.606
Wikidata*	600	0	130	7	-0.701
Wikidata*	600	1	125	5	-0.724
Wikidata*	600	2	127	6	-0.625
Wikidata*	600	3	128	6	-0.583
Wikidata*	600	4	127	6	-0.583
Wikidata*	1,200	0	301	7	-0.430
Wikidata*	1,200	1	302	5	-0.430
Wikidata*	1,200	2	293	10	-0.471
Wikidata*	1,200	3	298	8	-0.488
Wikidata*	1,200	4	289	11	-0.422

The real-data LLM calls contain 20 new records and the current node descriptions. They do not contain the full raw history. Qwen2.5-14B-Instruct uses its chat template, greedy decoding, and up to 1,400 new tokens. The Gemini results record the served identifier `gemini-3.8-flash`; the runner uses temperature zero and up to 4,096 output tokens. This runner does not explicitly set a low reasoning effort; that setting belongs to the separate archived depth protocol. We report the saved identifier, not a claim about public model availability or a substitute inferred from a runner default.

The parser tries to recover a JSON array, with an object-level fallback. Invalid record IDs and answers lacking a recognized node are skipped; background records remain ungrouped. The parser checks IDs against the prefix rather than enforcing that they occur in the current chunk. The real-data job summaries retain answer counts and aggregate results, but not full historical request/reply traces.

Therefore precise per-record omission, duplication, and repair rates cannot be reconstructed reliably. Parsed-object totals are not valid unique-record coverage: duplicate IDs and IDs from earlier chunks can be counted. This is a baseline-validation gap, not evidence by itself that the entire harness was broken.

Table 32: Every zero-shot real-data LLM arm. Parsed counts recovered answer objects across calls; proposed counts eligible groups entering certification. Selection is retained, rejected after eligible proposals, or no eligible group. Counts do not establish why a record was left ungrouped.

Corpus	n	Model	Parsed	Proposed	Certified	Selection
Amazon	600	Qwen	600	24	3	retained
Amazon	600	Gemini	311	15	0	rejected
Amazon	1,200	Qwen	1200	29	3	retained
Amazon	1,200	Gemini	749	30	5	retained
Amazon*	600	Qwen	598	18	2	retained
Amazon*	600	Gemini	538	26	5	retained
Amazon*	1,200	Qwen	1197	20	3	retained
Amazon*	1,200	Gemini	1135	33	15	retained
Wikidata	600	Qwen	600	0	0	no eligible group
Wikidata	600	Gemini	156	2	0	rejected
Wikidata	1,200	Qwen	1200	1	0	rejected
Wikidata	1,200	Gemini	455	14	3	retained
Wikidata*	600	Qwen	600	0	0	no eligible group
Wikidata*	600	Gemini	193	1	0	rejected
Wikidata*	1,200	Qwen	1200	1	0	rejected
Wikidata*	1,200	Gemini	329	1	0	rejected

Eight of sixteen arms have zero certified groups, including seven of eight Wikidata arms. Qwen yields 20 parsed objects per call in all four Wikidata conditions, with only `person` in the saved name list. The eligible proposal counts are 0, 1, 0, and 1, and all are rejected or absent. The adapter excludes groups with fewer than two members and groups spanning the entire prefix. Coarse grouping is a plausible explanation, but exact historical assignments are not saved to establish it.

Gemini yields 156, 455, 193, and 329 parsed objects for the corresponding Wikidata prefixes of 600, 1,200, 600, and 1,200 records. Three saved name lists mainly contain individual people; the unmodified 1,200-record list has 14 occupation names and retains three groups. The summaries therefore suggest both overly fine and overly coarse proposals, alongside incomplete parsed output. The matched study in Appendix D logs every request, response, assignment, and graph. Those complete traces support its coverage and future-block comparisons. They cannot reconstruct the unsaved memberships of this separate single-block study.

The three-example experiment changes only the demonstrated prompt prefix on Amazon 600. All examples and the zero-shot template are printed in Appendix H. Table 33 reports every available cell.

Table 33: Single-block Amazon zero-shot and few-shot comparison, prefix 600. ESCROW scores 1.454 bits per record on this input. Certificates are prefix scores; predictive bits are measured on the next block. These results use the single-block protocol and output allowances, separately from the matched Wikidata study.

Model	Prompt	Bits / rec.	Proposed	Certified	Certificate
Qwen	Zero	0.276	24	3	898.8
Qwen	Few	-0.058	40	3	207.9
Gemini	Zero	-0.269	15	0	-1.0
Gemini	Few	-0.269	15	0	7.1

Qwen forms certified groups in both conditions; Gemini fails the 10-bit margin in both. Neither finding proves inability across model families, prompts, or larger inference budgets.

E.5 SEMANTIC AND MEMORY HEAD COMPONENT STUDY

The saved R27 study uses the fixed prediction pipeline in Appendix C.5. Synthetic cases have a 1,200-record prefix and one 100-record block, at depths 2, 4, and 6 with seeds 34000 and 34001, each with a matched null. Seven real cases use the listed 600- or 1,200-record prefixes and one 100-record block. Amazon without product type at prefix 600 is absent from this component study and is not imputed. All 19 saved cases are included.

The semantic comparison holds the selected graph fixed. Both arms mix normalized experts with sharing rate .01; the full arm adds the within-group shared-count expert. Removing the head changes a three-expert mixture into a two-expert mixture, including its initialization cost. The group-only arm sets `hedge=False`: it uses the ordinary group expert and disables access to the semantic expert as well as weight adaptation. Its difference from the full arm is a joint removal, not an isolated estimate of memory, forgetting, or cross-block persistence. All arms here reset weights for their single scored block.

Table 34: Every head-component outcome, conditional bits saved per record. "No semantic" retains the memory head with two experts. "Group only" removes mixing and semantic prediction jointly. N is the number of certified groups, held fixed for these component removals. Full and empty scores agree on all six nulls.

Case	Prefix	Full	No semantic	Group only	Empty	N
abo_600	600	1.4543	1.1624	-1.7745	-0.2691	11
abo_1200	1200	1.7720	1.1527	-1.7690	-0.2555	17
abo_noptype_1200	1200	0.8293	0.5315	-2.7385	-0.2464	15
wikidata_600	600	0.0205	-0.5635	-42.2746	-1.2428	14
wikidata_1200	1200	0.0809	-0.4544	-23.3667	-0.9179	30
wikidata_noleak_600	600	-0.5231	-0.8774	-41.2276	-1.1160	11
wikidata_noleak_1200	1200	0.0444	-0.3098	-21.4989	-0.7112	25
d2_s34000_n1300	1200	6.4425	6.4463	5.6317	-0.0500	5
d2_s34001_n1300	1200	6.1323	5.9775	5.3252	-0.0740	6
d4_s34000_n1300	1200	21.3686	21.4343	20.0789	-0.0411	11
d4_s34001_n1300	1200	21.2216	21.2133	20.1013	-0.1406	11
d6_s34000_n1300	1200	39.3206	39.3955	37.5065	-0.0699	19
d6_s34001_n1300	1200	38.7373	38.6533	37.3072	-0.1275	26
d2_s34000_n1300_null	1200	-0.0285	-0.0163	-0.1481	-0.0285	0
d2_s34001_n1300_null	1200	0.0085	0.0104	-0.1132	0.0085	0
d4_s34000_n1300_null	1200	-0.0752	-0.0609	-0.3526	-0.0752	0
d4_s34001_n1300_null	1200	-0.0051	0.0008	-0.2335	-0.0051	0
d6_s34000_n1300_null	1200	-0.0906	-0.0897	-0.6257	-0.0906	0
d6_s34001_n1300_null	1200	-0.1276	-0.1136	-0.6947	-0.1276	0

Semantic sharing improves all seven real cases and three of six synthetic signal cases. The full predictor exceeds group-only prediction on all 13 signal cases. All six synthetic nulls retain zero groups. One null has a small positive block score; a positive increment alone is not a threshold crossing. These are descriptive paired results from the stated cases, not independent estimates across 13 domains.

E.6 MECHANISMS, ORDINARY PREDICTION, AND COST

T9 asks whether fresh evidence accumulates on signal and remains controlled on nulls. Depths are 1, 2, 3, 4, and 6, with seeds 34000 and 34001. Each of the ten signal streams has a paired null. After 200 warmup records, ten fresh blocks of 100 records are scored. The rate divides final evidence by 1,000 scored records. The initial log-wealth zero is included in the maximum.

Table 35: Every accumulated-evidence run. A negative final value is a loss of conditional evidence, not a negative probability.

Depth	Seed	Stream	Final bits	Max incl. start	Bits / scored rec.
1	34000	signal	-43.9	0.0	-0.044
1	34000	null	-497.6	0.0	-0.498
1	34001	signal	119.9	119.9	0.120
1	34001	null	-548.1	0.0	-0.548
2	34000	signal	4764.5	4764.5	4.765
2	34000	null	-560.1	0.0	-0.560
2	34001	signal	4798.6	4798.6	4.799
2	34001	null	-621.0	0.0	-0.621
3	34000	signal	11080.5	11080.5	11.080
3	34000	null	-677.4	0.0	-0.677
3	34001	signal	11052.7	11052.7	11.053
3	34001	null	-719.6	0.0	-0.720
4	34000	signal	18593.7	18593.7	18.594
4	34000	null	-721.9	0.0	-0.722
4	34001	signal	18865.2	18865.2	18.865
4	34001	null	-788.2	0.0	-0.788
6	34000	signal	33725.9	33725.9	33.726
6	34000	null	-1563.0	0.0	-1.563
6	34001	signal	33002.8	33002.8	33.003
6	34001	null	-1357.4	0.0	-1.357

The nuisance-field study fixes each declared depth and changes the number of irrelevant fields. It measures ordinary predictive saving for the proposed pool, COBWEB, and a supplied reference pool. It has one stream per condition, so it identifies a failure mechanism rather than a population effect estimate.

Table 36: Field-sensitivity study. Values are predictive bits saved per record.

Depth	Useful fields	Noise fields	ESCROW	COBWEB	Oracle pool
1	3	0	1.867	1.847	1.983
1	3	2	1.595	1.731	1.865
1	3	4	1.099	1.507	1.755
1	3	6	0.686	1.304	1.550
2	6	0	6.763	6.251	6.202
2	6	2	6.314	5.908	5.908
2	6	4	5.914	5.638	5.638
2	6	6	5.432	5.350	5.350
3	9	0	13.203	12.768	13.163
3	9	2	13.221	12.260	12.673
3	9	4	12.666	12.079	12.301
3	9	6	12.258	11.135	11.867
4	12	0	22.125	21.161	21.872
4	12	2	21.268	20.700	21.421
4	12	4	21.462	20.246	21.051
4	12	6	20.836	19.764	20.637

T7 separately uses a 600-record prefix and a 600-record evaluation block, a shared symbol inventory, and a certifier margin of ten. The inspected runner then applies a final acceptance cutoff of $10 + \log_2(2 \cdot 3) = 12.5849625$ bits. None of the 120 saved arm certificates lies between these cutoffs, so the saved final decisions agree under either cutoff. T7 is a single-prefix comparison. Its ordinary predictive rate is relative to a frozen background predictor. It does not use the conditional block-type reference and is not an evidence process under Theorem 2. The complete table includes cascade, CodeCobweb proposals, parent-based proposals, ordinary COBWEB, the supplied truth, and random size-matched groups. Every arm is certified. CodeCobweb is a separate entropy-based generator, not ordinary COBWEB with certification added. It scores lower than certified ordinary COBWEB in

all ten signal cases, by 4.305 bits per record on average. At depth one, seed 34000, their candidate pools have 76 and 245 groups, respectively; their selected graphs have zero and 16 groups. The supplied reference retains both of its two groups under the same certifier. These results expose a weaker proposal pipeline but do not isolate the causal effect of pruning. Appendix E.7 addresses that question using identical FISHDBC candidates.

Table 37: T7 ordinary predictive bits saved per record. CodeCobweb and ordinary COBWEB are distinct proposers using the same certifier; "Parent" is the parent-based proposer. A dagger marks an empty selected graph. The following table gives its certified group counts.

Depth	Seed	Stream	Cascade	CodeCobweb	Parent	COBWEB	Truth	Shape
1	34000	signal	1.388	0.000†	0.662	1.304	1.550	0.000†
1	34000	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
1	34001	signal	1.312	0.000†	0.850	1.371	1.591	0.000†
1	34001	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
2	34000	signal	5.350	1.771	5.429	5.350	5.350	0.000†
2	34000	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
2	34001	signal	5.316	2.578	5.338	5.316	5.316	0.000†
2	34001	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
3	34000	signal	11.715	10.440	12.258	11.906	11.867	0.000†
3	34000	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
3	34001	signal	11.861	7.874	12.319	11.725	12.008	0.000†
3	34001	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
4	34000	signal	19.535	18.007	20.836	19.925	20.637	0.000†
4	34000	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
4	34001	signal	19.780	16.930	21.158	20.146	20.782	0.000†
4	34001	null	0.000†	0.000†	0.000†	0.000†	0.000†	0.000†
6	34000	signal	35.971	25.174	38.893	37.816	37.585	−0.000†
6	34000	null	−0.000†	−0.000†	−0.000†	−0.000†	−0.000†	−0.000†
6	34001	signal	34.274	26.317	37.892	37.282	37.645	−0.000†
6	34001	null	−0.000†	−0.000†	−0.000†	−0.000†	−0.000†	−0.000†

Table 38: Certified nodes for every T7 score, in exactly the same row and column order. An approximately zero score with zero nodes is the background predictor, not evidence that a nonempty proposed hierarchy matched the background's performance.

Depth	Seed	Stream	Cascade	CodeCobweb	Parent	COBWEB	Truth	Shape
1	34000	signal	4	0	6	16	2	0
1	34000	null	0	0	0	0	0	0
1	34001	signal	3	0	5	17	2	0
1	34001	null	0	0	0	0	0	0
2	34000	signal	6	4	5	6	6	0
2	34000	null	0	0	0	0	0	0
2	34001	signal	6	4	6	6	6	0
2	34001	null	0	0	0	0	0	0
3	34000	signal	13	10	7	16	14	0
3	34000	null	0	0	0	0	0	0
3	34001	signal	13	6	7	14	14	0
3	34001	null	0	0	0	0	0	0
4	34000	signal	14	9	11	19	30	0
4	34000	null	0	0	0	0	0	0
4	34001	signal	14	9	11	20	30	0
4	34001	null	0	0	0	0	0	0
6	34000	signal	22	15	19	31	102	0
6	34000	null	0	0	0	0	0	0
6	34001	signal	11	13	24	34	96	0
6	34001	null	0	0	0	0	0	0

Table 39: Single-block real-data component removals. Missing Amazon* 600 results are not replaced with another run. “No hedge” selects the unsmoothed group expert, so it removes both expert mixing and access to the separate sharing expert.

Case	Full	No sharing	No hedge	Empty
abo_600	1.454	1.162	-1.775	-0.269
abo_1200	1.772	1.153	-1.769	-0.256
abo_noptype_1200	0.829	0.532	-2.738	-0.246
wikidata_600	0.020	-0.564	-42.275	-1.243
wikidata_1200	0.081	-0.454	-23.367	-0.918
wikidata_noleak_600	-0.523	-0.877	-41.228	-1.116
wikidata_noleak_1200	0.044	-0.310	-21.499	-0.711

The last ablation is therefore a composite intervention, not an isolated estimate of the effect of weight updates. The full versus no-sharing comparison is the cleaner comparison for the similarity component.

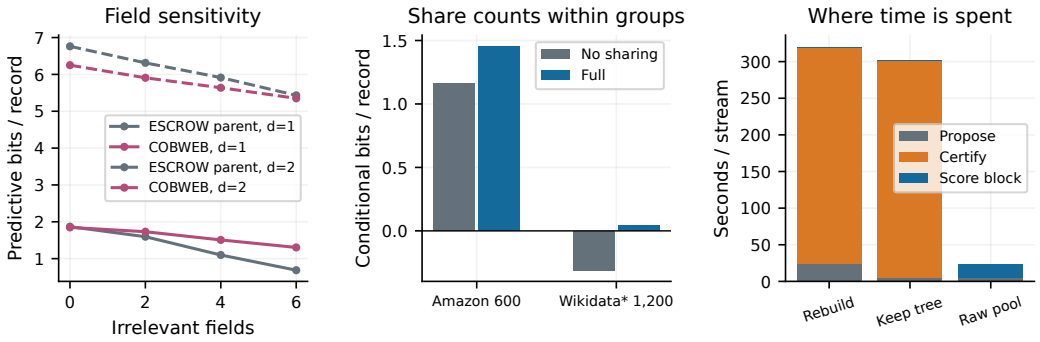


Figure 12: Supporting component diagnostics. Left: nuisance fields, one stream per condition. Middle: semantic count sharing on fixed groups. Right: cost over three signal streams. These studies have separate protocols and are not a combined ablation.

T10 uses depths 2, 3, and 4 at seed 34000, signal and matched null, with the same 1,200-record length and block schedule. Arm A rebuilds and certifies, B keeps the proposal tree but certifies at repairs, and C keeps the raw pool without repeated certification. Times include generation, certification, and block scoring. They are observed run times, not repeated hardware-controlled latency estimates.

Table 40: Every saved cost row, seconds per stream. Evidence is accumulated bits. Nodes is the mean selected or raw node count across ten repair checkpoints, not a terminal count. On nulls, A and B keep zero groups while C holds 167.4 to 189.1 on average. Arm C changes the supported output contract as well as the running time.

Case	Arm	Propose	Certify	Score	Total (s)	Evidence	Nodes
d2 signal	A	21.0	240.0	0.7	261.7	4765	4.9
d2 signal	B	4.2	239.9	0.7	244.8	4765	4.9
d2 signal	C	4.1	0.6	14.2	18.9	4370	182.0
d2 null	A	26.9	262.1	0.3	289.4	-560	0.0
d2 null	B	5.1	262.4	0.3	267.9	-560	0.0
d2 null	C	5.1	0.6	13.8	19.5	-183	175.5
d3 signal	A	22.9	278.2	1.0	302.1	11080	7.0
d3 signal	B	4.6	281.1	1.0	286.7	11080	7.0
d3 signal	C	4.6	0.8	18.6	24.0	9804	198.0
d3 null	A	43.3	264.1	0.4	307.8	-677	0.0
d3 null	B	8.4	264.5	0.4	273.2	-677	0.0
d3 null	C	8.3	0.5	15.9	24.7	-306	167.4
d4 signal	A	27.5	367.6	1.7	396.8	18594	10.8
d4 signal	B	5.3	368.1	1.7	375.1	18594	10.8
d4 signal	C	5.3	0.7	23.0	29.0	16975	199.4
d4 null	A	86.6	423.3	0.5	510.4	-722	0.0
d4 null	B	19.4	427.6	0.5	447.5	-722	0.0
d4 null	C	19.1	0.6	22.0	41.7	-225	189.1

How many prefix tests were made? Each real-data run certifies once after its declared prefix. Thirty or sixty chunked LLM calls do not imply thirty or sixty certification tests. T7 also certifies once. T9 and T10 make decisions at prefixes 200, 300, up to 1,100, giving ten checks. If the alpha-spending rule in Appendix B were used with test index $j = 1, \ldots, 10$, its last margin would be $10 + \log_2(10 \cdot 11) = 16.7814$ bits. That rule was not used for these saved repair decisions. Their summaries do not save each checkpoint's certificate, so unchanged acceptance cannot be established from the published aggregates. A replay must also allow changed decisions to affect later state. The reported fresh-block wealth has a separate global-null guarantee and does not repair the missing repeated-prefix structural guarantee.

E.7 PRUNING FIXED PROPOSAL POOLS

Question and design. Does pruning rescue a useful part of a costly graph, or remove groups that help predict new records? T7 changes the proposer and cannot isolate this effect. Here all arms start with the same candidate memberships and parent links. We freeze the 42 native FISHDBC graphs from the depth study: depths one through seven, seeds 34000 through 34002, and one signal stream and its matched null at each setting. Each graph uses 300 records. The conditional arms use the paper's frozen certifier and optional flagged value code. This is an exploratory diagnostic on reused development streams. Its protocol and source hashes were frozen before computing the outcomes; it is not independent confirmation or an externally preregistered study.

Three arms, one candidate pool. The native arm retains the archived memberships and links without a support threshold. The whole-certificate arm installs those same groups, chooses field supports and ownership, and applies the conditional 10-bit threshold with pruning disabled. The pruned-certificate arm uses the same installation and support search, then runs the conditional node-deletion routine before applying that threshold. The initial installed scores are checked to be identical between certified arms. This comparison includes support changes caused by deletion and the final threshold decision, as those are parts of the pruning procedure. It does not isolate each of these steps individually.

The whole conditional-certificate arm is distinct from the archived EHR05 replacement filter. It tests one fixed prefix using the conditional code, not several successive EHR05 replacements. Native graphs are reindexed for the conditional implementation, with membership and link preservation checked. Their recomputed recovery metrics equal the historical native metrics. No FISHDBC or LLM proposer is retrained or called.

A fresh block for prediction. We extend each signal fixture to 400 records and verify that its first 300 are unchanged. Records 301 through 400 form one evaluation block. For a null case, the prefix remains the exact archived column-shuffled prefix. Its future block is a separately shuffled copy of the matched signal continuation, preserving each column's counts. The declared shuffle seed is $991000 + \text{seed}$. Prefix hashes match the archived protocol in all 42 cases. Future records and reference labels do not enter graph selection.

The primary analysis scores the explicit nodes in `result['state']`, not the additional residual groups available in `result['groups']`. All three graphs use the same frozen ordinary mixture predictor from T7. Its score is bits saved per future record against the frozen background predictor. Alphabet sizes are shared across arms and computed from the prefix and evaluation block, as in T7. It fits every field in each explicit group, ignoring selected field supports, ownership, and parent links except for traversal order. This disclosed alphabet access means the experiment is not a predictor with no knowledge of the evaluation vocabulary. These scores are not the conditional wealth of Theorem 2. Reference memberships and links are used only for evaluation.

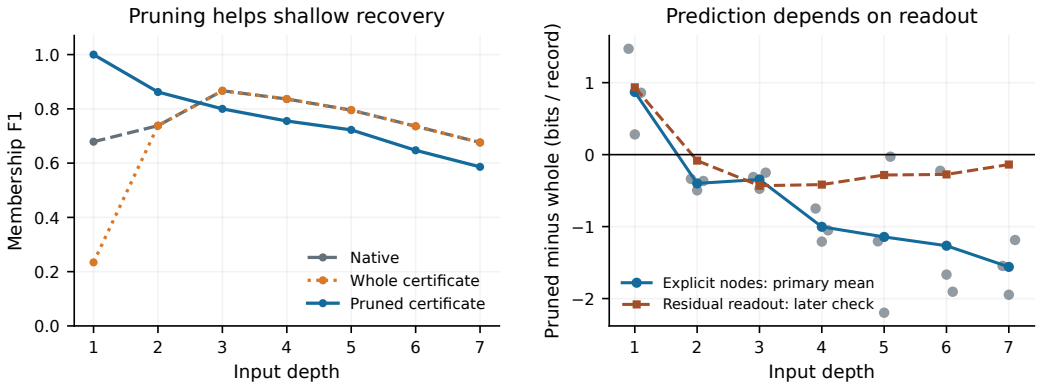


Figure 13: Same-pool comparison, three signal streams per depth. Left: mean membership F1. Native and whole-certificate curves overlap above depth one. Right: paired predictive change from enabling pruning; dots are the three streams, horizontally offset for visibility, and the solid line is their mean for explicit nodes. The dashed line adds residual groups in a post-result sensitivity check. These are not confidence intervals. Both certified arms select zero groups on all 21 matched nulls.

Outcomes and their meaning. The whole-certificate arm accepts 19 of 21 signal graphs; the pruned arm accepts all 21. Both select no groups on all 21 null streams. At depth one, pruning returns exactly the two reference groups on every stream, with no false links. Whole-graph certification instead rejects two candidates and retains one graph with six false parent links. Pruning can therefore rescue useful structure that a whole-graph decision discards.

The benefit is not uniform. Membership and relationship F1 improve at depth two but decrease on all 15 streams at depths three through seven. Prediction improves on all three depth-one streams relative to the whole-certificate arm, then decreases on all 18 deeper streams. The mean predictive changes at depths two through seven are -0.400, -0.345, -1.003, -1.143, -1.266, and -1.560 bits per record. Their pooled mean is -0.953; each depth has three streams, with more fields at greater depth. The pooled change over all 21 signal streams is -0.692. Thus a better support score need not give better explicit-node recovery or this ordinary mixture score.

These data establish a cost under this explicit-node evaluation convention. The later audit below finds that the size of this cost depends strongly on group readout. The results do not show that every code or pruning rule has that cost, or that every deleted group was useful. The search optimizes its declared code objective, which differs from future predictive loss, and is greedy rather than globally optimal. A node-deletion procedure also does not establish a separate error guarantee for each retained node. Further work should distinguish proposal quality, the search objective, and finite-sample group costs without choosing settings on these reported outcomes.

Candidate counts do not establish the cause. At depths four through seven the native pool already has fewer groups than the reference. This may limit repair by deletion. Yet at depth three the pool has 18 groups for 14 reference groups, and pruning still lowers recovery in all three streams. Deleting a false group can improve precision even when other groups are missing. The average removed fraction at depths two through seven is 57.98%, with a wide 33.33% to 83.33% range. At depth one the mean membership-F1 gain is 0.766 relative to whole-graph certification, but 0.321 relative to native proposals: the larger gain includes rescue from rejection. In the four traced cases, 30 of 55 deletions improve the ordinary prediction score and 25 lower it. The scorer does not penalize every deletion. Also, depth and field count increase together, so this design cannot separate their effects. Appendix E.8 varies candidate supply at fixed depth and field count without assuming that more proposals are better.

Table 41: Same-pool signal means over three streams per row. Whole and Pruned use the same conditional code and threshold. Bits / rec. is the ordinary future-block score, not a certificate. Depth-one relation F1 is n/a because the reference has no links.

Depth	Arm	Nodes	Member F1	Parent F1	Anc. F1	Bits / rec.
1	Native	10.00	0.679	n/a	n/a	1.249
1	Whole	2.67	0.234	n/a	n/a	0.441
1	Pruned	2.00	1.000	n/a	n/a	1.311
2	Native	20.67	0.738	0.204	0.179	5.167
2	Whole	20.67	0.738	0.204	0.179	5.167
2	Pruned	4.33	0.862	0.730	0.730	4.767
3	Native	18.00	0.867	0.839	0.752	10.907
3	Whole	18.00	0.867	0.839	0.752	10.907
3	Pruned	6.00	0.800	0.500	0.333	10.561
4	Native	18.67	0.836	0.686	0.666	18.041
4	Whole	18.67	0.836	0.686	0.666	18.041
4	Pruned	9.33	0.755	0.413	0.269	17.038
5	Native	24.67	0.795	0.449	0.423	25.291
5	Whole	24.67	0.795	0.449	0.423	25.291
5	Pruned	12.67	0.722	0.301	0.165	24.148
6	Native	30.67	0.736	0.267	0.248	30.512
6	Whole	30.67	0.736	0.267	0.248	30.512
6	Pruned	14.33	0.647	0.167	0.083	29.247
7	Native	31.33	0.676	0.167	0.119	35.414
7	Whole	31.33	0.676	0.167	0.119	35.414
7	Pruned	14.33	0.586	0.088	0.036	33.854

Complete outcomes and reproduction. The tables beginning with Tables 42 and 45 retain all 126 arm outcomes. Certificate columns give the score before the final threshold replaces a graph with the empty graph; native outputs have no certificate in this comparison. Zero-node predictors remain in every comparison. Tiny floating-point differences from zero are printed to three decimals. Saved JSON files retain complete graphs, pre-threshold graphs, owners, individual deletion steps, certificates, predictions, and source hashes.

Run `scripts/pruning_comparison.py --run` with NumPy and SciPy to reproduce from frozen inputs. The optional `--prepare` step needs the original archive and must not overwrite an existing frozen protocol. The local execution used two workers, Python 3.12.12, NumPy 2.2.6, and SciPy 1.15.3. An initial launch lacked SciPy and stopped during native-metric validation before any arm outcome; the unchanged protocol then completed in the existing environment. All 42 cases finished. Runtime measurements describe this execution only and are not a controlled speed benchmark.

Table 42: Every signal outcome for seed 34000 in the same-pool follow-up. Relation F1 is n/a at depth one. Empty selected graphs retain their scores.

Depth	Seed	Arm	Nodes	Member F1	Par. F1	Anc. F1	Cert.	Bits / rec.
1	34000	Native	12	0.693	n/a	n/a	–	1.558
1	34000	Whole	0	0.000	n/a	n/a	-284.5	0.000
1	34000	Pruned	2	1.000	n/a	n/a	407.8	1.471
2	34000	Native	16	0.847	0.444	0.250	–	5.366
2	34000	Whole	16	0.847	0.444	0.250	668.7	5.366
2	34000	Pruned	5	0.915	0.857	0.857	1291.9	5.030
3	34000	Native	20	0.857	0.800	0.667	–	10.987
3	34000	Whole	20	0.857	0.800	0.667	2039.1	10.987
3	34000	Pruned	6	0.800	0.500	0.333	2741.6	10.676
4	34000	Native	18	0.810	0.682	0.640	–	17.502
4	34000	Whole	18	0.810	0.682	0.640	3358.6	17.502
4	34000	Pruned	8	0.715	0.353	0.211	3868.9	16.753
5	34000	Native	28	0.806	0.419	0.481	–	25.238
5	34000	Whole	28	0.806	0.419	0.481	4464.3	25.238
5	34000	Pruned	11	0.692	0.261	0.133	5142.4	24.033
6	34000	Native	30	0.725	0.252	0.241	–	30.517
6	34000	Whole	30	0.725	0.252	0.241	4438.5	30.517
6	34000	Pruned	14	0.657	0.163	0.075	5088.2	30.292
7	34000	Native	34	0.680	0.148	0.133	–	35.355
7	34000	Whole	34	0.680	0.148	0.133	4416.0	35.355
7	34000	Pruned	14	0.576	0.080	0.034	4956.8	33.809

Table 43: Every signal outcome for seed 34001 in the same-pool follow-up. Relation F1 is n/a at depth one. Empty selected graphs retain their scores.

Depth	Seed	Arm	Nodes	Member F1	Par. F1	Anc. F1	Cert.	Bits / rec.
1	34001	Native	8	0.703	n/a	n/a	–	1.322
1	34001	Whole	8	0.703	n/a	n/a	29.4	1.322
1	34001	Pruned	2	1.000	n/a	n/a	408.0	1.603
2	34001	Native	22	0.726	0.167	0.154	–	5.433
2	34001	Whole	22	0.726	0.167	0.154	337.5	5.433
2	34001	Pruned	4	0.833	0.667	0.667	1322.5	4.937
3	34001	Native	20	0.880	0.800	0.690	–	11.081
3	34001	Whole	20	0.880	0.800	0.690	2033.4	11.081
3	34001	Pruned	6	0.800	0.500	0.333	2728.6	10.607
4	34001	Native	18	0.833	0.636	0.640	–	18.356
4	34001	Whole	18	0.833	0.636	0.640	3723.1	18.356
4	34001	Pruned	9	0.763	0.400	0.256	4091.8	17.148
5	34001	Native	18	0.762	0.395	0.296	–	25.390
5	34001	Whole	18	0.762	0.395	0.296	5002.5	25.390
5	34001	Pruned	12	0.715	0.286	0.151	5229.1	23.193
6	34001	Native	32	0.753	0.312	0.263	–	30.817
6	34001	Whole	32	0.753	0.312	0.263	4457.2	30.817
6	34001	Pruned	16	0.655	0.188	0.103	4988.2	29.150
7	34001	Native	32	0.682	0.185	0.122	–	35.532
7	34001	Whole	32	0.682	0.185	0.122	3940.9	35.532
7	34001	Pruned	14	0.590	0.087	0.034	4579.5	33.585

Table 44: Every signal outcome for seed 34002 in the same-pool follow-up. Relation F1 is n/a at depth one. Empty selected graphs retain their scores.

Depth	Seed	Arm	Nodes	Member F1	Par. F1	Anc. F1	Cert.	Bits / rec.
1	34002	Native	10	0.641	n/a	n/a	–	0.867
1	34002	Whole	0	0.000	n/a	n/a	-164.3	0.000
1	34002	Pruned	2	1.000	n/a	n/a	406.8	0.860
2	34002	Native	24	0.641	0.000	0.133	–	4.701
2	34002	Whole	24	0.641	0.000	0.133	193.6	4.701
2	34002	Pruned	4	0.838	0.667	0.667	1218.9	4.335
3	34002	Native	14	0.863	0.917	0.900	–	10.651
3	34002	Whole	14	0.863	0.917	0.900	2223.6	10.651
3	34002	Pruned	6	0.800	0.500	0.333	2725.1	10.401
4	34002	Native	20	0.866	0.739	0.717	–	18.264
4	34002	Whole	20	0.866	0.739	0.717	4074.2	18.264
4	34002	Pruned	11	0.788	0.486	0.341	4470.1	17.212
5	34002	Native	28	0.818	0.535	0.492	–	25.246
5	34002	Whole	28	0.818	0.535	0.492	4766.9	25.246
5	34002	Pruned	15	0.761	0.356	0.210	5275.9	25.217
6	34002	Native	30	0.730	0.238	0.241	–	30.203
6	34002	Whole	30	0.730	0.238	0.241	4497.8	30.203
6	34002	Pruned	13	0.630	0.149	0.072	5154.7	28.298
7	34002	Native	28	0.667	0.169	0.103	–	35.356
7	34002	Whole	28	0.667	0.169	0.103	4411.3	35.356
7	34002	Pruned	15	0.593	0.097	0.040	4919.7	34.169

Table 45: Every matched null outcome for seed 34000 in the same-pool follow-up. All returned links are false.

Depth	Seed	Arm	Nodes	False par.	False anc.	Cert.	Bits / rec.
1	34000	Native	20	18	84	–	-0.080
1	34000	Whole	0	0	0	-1241.5	0.000
1	34000	Pruned	0	0	0	-1.0	0.000
2	34000	Native	14	12	42	–	-0.459
2	34000	Whole	0	0	0	-956.8	0.000
2	34000	Pruned	0	0	0	-1.0	0.000
3	34000	Native	16	14	48	–	-0.193
3	34000	Whole	0	0	0	-1056.9	0.000
3	34000	Pruned	0	0	0	-1.0	0.000
4	34000	Native	8	6	12	–	-0.610
4	34000	Whole	0	0	0	-785.1	0.000
4	34000	Pruned	0	0	0	-1.0	0.000
5	34000	Native	12	10	30	–	-0.661
5	34000	Whole	0	0	0	-713.8	0.000
5	34000	Pruned	0	0	0	-1.0	0.000
6	34000	Native	6	4	6	–	-0.442
6	34000	Whole	0	0	0	-556.0	-0.000
6	34000	Pruned	0	0	0	-1.0	-0.000
7	34000	Native	10	8	20	–	0.324
7	34000	Whole	0	0	0	-923.9	-0.000
7	34000	Pruned	0	0	0	-1.0	-0.000

Table 46: Every matched null outcome for seed 34001 in the same-pool follow-up. All returned links are false.

Depth	Seed	Arm	Nodes	False par.	False anc.	Cert.	Bits / rec.
1	34001	Native	16	14	42	–	-0.087
1	34001	Whole	0	0	0	-1070.5	0.000
1	34001	Pruned	0	0	0	-1.0	0.000
2	34001	Native	14	12	40	–	-0.174
2	34001	Whole	0	0	0	-955.2	0.000
2	34001	Pruned	0	0	0	-1.0	0.000
3	34001	Native	10	8	20	–	-0.347
3	34001	Whole	0	0	0	-819.6	0.000
3	34001	Pruned	0	0	0	-1.0	0.000
4	34001	Native	14	12	38	–	-0.469
4	34001	Whole	0	0	0	-922.0	0.000
4	34001	Pruned	0	0	0	-1.0	0.000
5	34001	Native	14	12	42	–	-0.544
5	34001	Whole	0	0	0	-899.6	0.000
5	34001	Pruned	0	0	0	-1.0	0.000
6	34001	Native	8	6	12	–	-0.292
6	34001	Whole	0	0	0	-576.9	-0.000
6	34001	Pruned	0	0	0	-1.0	-0.000
7	34001	Native	12	10	30	–	0.537
7	34001	Whole	0	0	0	-764.8	-0.000
7	34001	Pruned	0	0	0	-1.0	-0.000

Table 47: Every matched null outcome for seed 34002 in the same-pool follow-up. All returned links are false.

Depth	Seed	Arm	Nodes	False par.	False anc.	Cert.	Bits / rec.
1	34002	Native	18	16	60	–	-0.486
1	34002	Whole	0	0	0	-1248.3	0.000
1	34002	Pruned	0	0	0	-1.0	0.000
2	34002	Native	16	14	44	–	-0.412
2	34002	Whole	0	0	0	-1087.0	0.000
2	34002	Pruned	0	0	0	-1.0	0.000
3	34002	Native	12	10	30	–	-0.589
3	34002	Whole	0	0	0	-847.6	0.000
3	34002	Pruned	0	0	0	-1.0	0.000
4	34002	Native	6	4	6	–	-0.304
4	34002	Whole	0	0	0	-459.4	0.000
4	34002	Pruned	0	0	0	-1.0	0.000
5	34002	Native	8	6	12	–	-0.635
5	34002	Whole	0	0	0	-610.6	0.000
5	34002	Pruned	0	0	0	-1.0	0.000
6	34002	Native	10	8	20	–	-0.289
6	34002	Whole	0	0	0	-581.9	-0.000
6	34002	Pruned	0	0	0	-1.0	-0.000
7	34002	Native	6	4	6	–	0.373
7	34002	Whole	0	0	0	-455.4	-0.000
7	34002	Pruned	0	0	0	-1.0	-0.000

E.7.1 Measurement audit and sensitivity

Independent arithmetic. After the first outcomes, we audited the surprising loss using a separate calculation of direct probability products and mixture sums. All 126 predictive rates agree within 5.12×10^{-12} bits per record. Exact rational enumeration on four small examples, including overlap, missing fields, and unseen values, verifies total probability one. Recovery scores reproduce from overlap counts and transitive closure. A separate assignment solver confirms the matching objectives. Fresh certifiers in reverse arm order reproduce all 84 certified outputs, excluding carryover between these arms.

What the score measures. The ordinary predictor fits every field in every explicit group. It does not use the certifier's field supports or ownership. The existing readout routine also exposes residual groups, such as a parent's members left after its supported children are removed, and rebuilds parent links from containment. Applying it adds memberships in 34 of the 84 certified outputs. These are two different evaluation conventions.

Keeping selection fixed, we scored that readout with the same ordinary predictor. The mean loss at depths two through seven falls from 0.953 to 0.271 bits per record; 17 of 18 streams lose, rather than all 18. Recovery still improves at depth two and decreases on all 15 deeper streams under this readout. This post-result diagnostic does not replace the frozen primary analysis. Neither convention in that audit tests the separate conditional shared-count predictor used on real data. Appendix E.8 adds that comparison under a separately frozen protocol.

Table 48: Mean predictive change from enabling pruning, bits per record. Three paired signal streams per depth. Loss counts compare pruned and whole-certificate outputs. Residual readout is a post-result sensitivity check, not a new primary endpoint.

Depth	Explicit-node change	Losses	Residual-readout change	Losses
1	0.871	0/3	0.935	0/3
2	-0.400	3/3	-0.083	3/3
3	-0.345	3/3	-0.432	3/3
4	-1.003	3/3	-0.417	3/3
5	-1.143	3/3	-0.283	3/3
6	-1.266	3/3	-0.275	3/3
7	-1.560	3/3	-0.138	2/3

Vocabulary and concrete traces. A separate diagnostic replaces evaluation-block vocabulary access with alphabet sizes known from the synthetic generator. All 18 deeper explicit-node losses remain, with mean 0.940 bits per record. Replaying every deletion at depths one, two, three, and seven for seed 34000 reproduces all 55 logged search savings. Vectorized and reference code lengths agree. For example, at depth seven, removing a 38-record node saves 27.785 search bits but costs 0.655 predictive bits per future record. That node supports no field in the selected code, yet the ordinary predictor uses all its field counts. This traces an objective mismatch; it does not prove one cause for every loss.

Matching ties and limits. At depth five, seed 34000, two optimal matchings for the native and whole-certificate graphs give membership F1 0.805654 versus 0.807224 and parent F1 0.418605 versus 0.441860. Prediction is unaffected. None of the three recovery comparisons changes direction under this observed alternative. These are two optimal solutions, not bounds over every tie. We retain the frozen matching and disclose the ambiguity. The audit supports arithmetic under the stated conventions, not every measurement choice or all possible inputs. The package includes `audit_measurements.py`, `trace_pruning_measurements.py`, `audit_readout_sensitivity.py`, and their complete JSON outputs under `evidence/pruning_comparison/`.

E.8 PARENT-BASED PROPOSALS ACROSS NOISE, DEPTH, AND REAL RECORDS

Purpose and status. This study tests parent-based insertion, the conditional certifier, and the shared-count predictor together. It does not use the recursive hierarchy search. It also revisits the pruning result using that predictor and varies candidate supply. The protocol, evaluation conventions, source hashes, input files, and runtime limits were frozen before computing new outcomes. The same-graph diagnostic reuses development inputs. The broader synthetic tests use fresh seeds; the real study draws new record orders from the same two available corpora. These are distinct sources of evidence, not independent datasets to be pooled.

The declared cases. There are 175 cases: 42 diagnostics on saved FISHDBC graphs, 42 fresh depth signal/null cases, 33 fresh noise-ladder cases, 40 real order/control cases, and 18 candidate-supply cases. Synthetic seeds are 61031, 61032, and 61033. The depth study has depths one through seven. The ladder uses signals 0, 0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.65, 0.80, 0.90, and 1.00. It follows the flat ladder's shared-vocabulary law with eight groups and six informative fields, then adds six independent irrelevant fields. Its 300-record prefix, extra fields, and current code differ from the 3,000-record classical ladder.

The real variants are Amazon with and without product type, and Wikidata with and without the previously identified hierarchy-revealing fields. For each variant, seeds 72001 through 72005 shuffle the full available corpus and take the first 900 records. Row indices and source hashes are saved. The paired variants use the same row orders. The proposer random seed also follows the order seed. Thus these runs vary the sample, its order, and COBWEB's randomness; they are not order-only perturbations of one fixed 900-record set. The corpora contain 19,352 and 3,362 records respectively, and different orders can reuse records.

Shared pipeline and information access. The benchmark compares the parent-based ESCROW proposer, COBWEB, FISHDBC, and an empty graph. Each nonempty candidate enters the same `FastCertifier` with `values=min`, pruning enabled, and margin 10. Generator defaults are fixed. FISHDBC uses minimum samples 5, neighborhood parameter 5, search parameter 50, balanced insertion, and minimum cluster size 5. It exports nonuniversal groups from its condensed hierarchy. COBWEB exports the ancestry of each prefix record's assigned concept. A proposer receives records and a random seed, never reference memberships, parent labels, field roles, or future records.

Each case has one structural decision: a 300-record synthetic prefix or a 600-record real prefix. No repeated-prefix support guarantee is claimed. The selected graph and its fitted counts remain fixed for three subsequent 100-record blocks. The diagnostic instead retains its original single 100-record block. Each block uses the current three-expert conditional scorer, sharing rate 0.01, and newly reset expert weights. Kernels use prefix co-occurrences only. The scorer knows the current block's column counts and field inventory, as required by its conditional reference.

The primary predictor uses explicit selected nodes. Both the ordinary and conditional predictors fit all fields within each component; selected supports and ownership do not directly gate the predictive counts. The conditional version adds the urn normalization, adaptive expert weights, and shared value counts. A separate residual readout adds owned row sets and rebuilds inclusion links. We declare that sensitivity before scoring this diagnostic; it does not replace the primary convention.

Nulls and evaluation. Every depth case has a paired column-shuffled null. Every real order has the same control. Shuffling includes absent cells and is independent by column, separately within the prefix and each future block. Each segment's column counts are checked to be unchanged. The zero-signal ladder supplies three further null cases. Recovery uses only prefix reference memberships. Null F1 and depth-one relationship F1 are not interpreted. The existing maximum summed group-F1 assignment is used, with sorted IDs and SciPy 1.15.3 recorded; selected matchings are saved.

Uncertainty and limits. For each real order we average its three block scores before comparing methods. The displayed 95% percentile intervals resample the five paired order averages 10,000 times, using seed 90323. They are pointwise descriptive intervals conditional on these corpora, with only five sampled orders. They do not treat 15 blocks as 15 independent datasets, account for new corpora, or justify a broad population claim. Synthetic means retain all three seeds. Field count still varies

with depth. This study evaluates fixed models on later blocks, not a complete continually updating streaming implementation. LLM and KNN comparisons use the matched protocol in Appendix D.

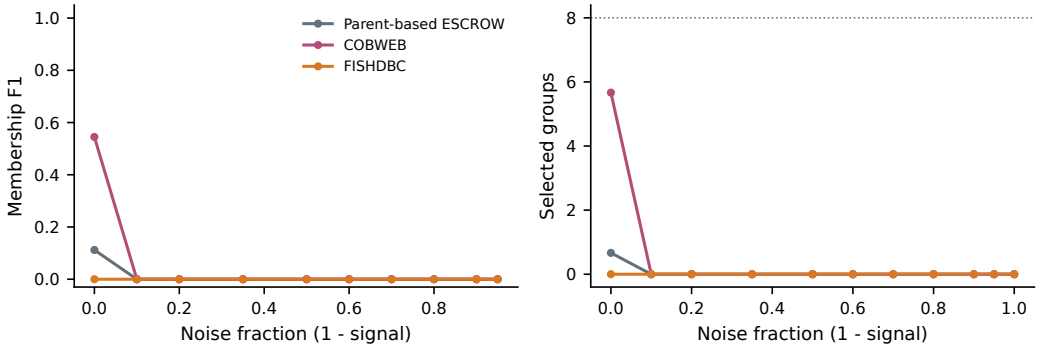


Figure 14: Parent-based short-stream stress test. Three seeds per level; 300 prefix records, six informative fields and six irrelevant fields. Every proposer uses the same conditional certifier. This differs from the flat-search classical ladder.

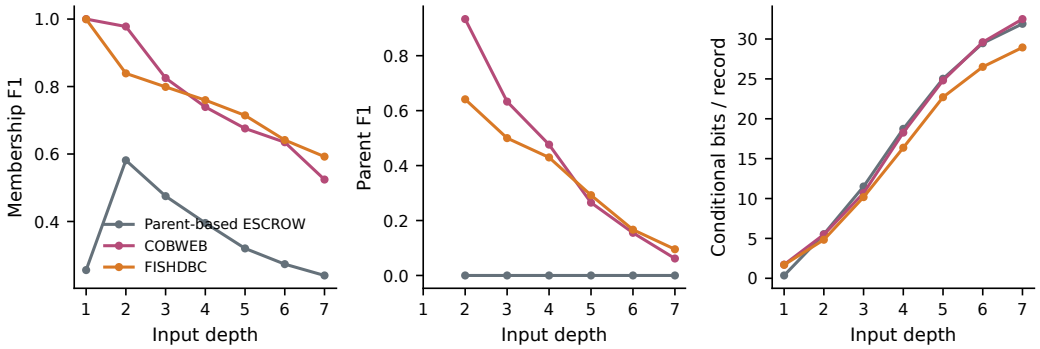


Figure 15: Parent-based pipeline on fresh depth streams, three seeds per depth. This figure measures that insertion proposer, not recursive ESCROW. Both baselines use the same conditional certifier and predictor. The same-input implementation comparison is in Appendix E.9.

Table 49: Parent-insertion noise ladder. Means over three streams per signal level. Bits per record average all three future blocks. No reference group is defined at signal zero.

Signal	Arm	Groups	Member F1	Bits / rec.
0	ESCROW-P	0.00	n/a	-0.155
0	COBWEB	0.00	n/a	-0.155
0	FISHDBC	0.00	n/a	-0.155
0.05	ESCROW-P	0.00	0.000	-0.157
0.05	COBWEB	0.00	0.000	-0.157
0.05	FISHDBC	0.00	0.000	-0.157
0.1	ESCROW-P	0.00	0.000	-0.156
0.1	COBWEB	0.00	0.000	-0.156
0.1	FISHDBC	0.00	0.000	-0.156
0.2	ESCROW-P	0.00	0.000	-0.160
0.2	COBWEB	0.00	0.000	-0.160
0.2	FISHDBC	0.00	0.000	-0.160
0.3	ESCROW-P	0.00	0.000	-0.163
0.3	COBWEB	0.00	0.000	-0.163
0.3	FISHDBC	0.00	0.000	-0.163
0.4	ESCROW-P	0.00	0.000	-0.163
0.4	COBWEB	0.00	0.000	-0.163
0.4	FISHDBC	0.00	0.000	-0.163
0.5	ESCROW-P	0.00	0.000	-0.164
0.5	COBWEB	0.00	0.000	-0.164
0.5	FISHDBC	0.00	0.000	-0.164
0.65	ESCROW-P	0.00	0.000	-0.167
0.65	COBWEB	0.00	0.000	-0.167
0.65	FISHDBC	0.00	0.000	-0.167
0.8	ESCROW-P	0.00	0.000	-0.161
0.8	COBWEB	0.00	0.000	-0.161
0.8	FISHDBC	0.00	0.000	-0.161
0.9	ESCROW-P	0.00	0.000	-0.161
0.9	COBWEB	0.00	0.000	-0.161
0.9	FISHDBC	0.00	0.000	-0.161
1	ESCROW-P	0.67	0.112	0.400
1	COBWEB	5.67	0.545	5.242
1	FISHDBC	0.00	0.000	-0.171

Table 50: Parent-insertion depth study. Three signal/null pairs per depth. Membership and links are evaluated on the 300-record prefix; prediction averages three later blocks.

Depth	Arm	Groups	Member F1	Parent F1	Anc. F1	Bits / rec.	Null groups
1	ESCROW-P	0.67	0.256	n/a	n/a	0.355	0.000
1	COBWEB	2.00	1.000	n/a	n/a	1.714	0.000
1	FISHDBC	2.00	0.999	n/a	n/a	1.687	0.000
2	ESCROW-P	6.33	0.581	0.000	0.000	5.530	0.000
2	COBWEB	6.67	0.978	0.933	0.889	5.490	0.000
2	FISHDBC	4.00	0.839	0.641	0.641	4.833	0.000
3	ESCROW-P	7.33	0.475	0.000	0.000	11.507	0.000
3	COBWEB	8.67	0.825	0.633	0.544	10.681	0.000
3	FISHDBC	6.00	0.799	0.500	0.333	10.177	0.000
4	ESCROW-P	10.33	0.396	0.000	0.019	18.736	0.000
4	COBWEB	12.67	0.739	0.476	0.326	18.264	0.000
4	FISHDBC	9.67	0.760	0.430	0.285	16.371	0.000
5	ESCROW-P	15.00	0.320	0.000	0.000	25.018	0.000
5	COBWEB	18.00	0.676	0.265	0.210	24.808	0.000
5	FISHDBC	12.67	0.714	0.292	0.165	22.706	0.000
6	ESCROW-P	18.67	0.274	0.000	0.000	29.469	0.000
6	COBWEB	25.33	0.635	0.155	0.145	29.601	0.000
6	FISHDBC	14.33	0.641	0.167	0.087	26.507	0.000
7	ESCROW-P	24.00	0.240	0.000	0.000	31.904	0.000
7	COBWEB	23.00	0.524	0.062	0.048	32.499	0.000
7	FISHDBC	15.33	0.592	0.095	0.042	28.940	0.000

Table 51: Parent-insertion real-data study. Each order score averages three blocks after one 600-record prefix. Min and max are observed order means, not confidence bounds. All five orders and every empty selected graph are retained.

Corpus	Arm	Mean	Min order	Max order	Mean groups
Amazon	Empty	-0.264	-0.282	-0.237	0.000
Amazon	ESCROW-P	1.434	1.126	1.642	7.800
Amazon	COBWEB	1.032	0.786	1.332	14.000
Amazon	FISHDBC	0.705	0.462	0.936	5.400
Amazon*	Empty	-0.241	-0.263	-0.216	0.000
Amazon*	ESCROW-P	0.753	0.592	1.049	8.400
Amazon*	COBWEB	0.376	-0.263	0.875	9.400
Amazon*	FISHDBC	0.093	-0.139	0.256	3.600
Wikidata	Empty	-1.527	-1.604	-1.420	0.000
Wikidata	ESCROW-P	-0.845	-1.030	-0.588	6.800
Wikidata	COBWEB	-1.371	-1.505	-1.258	2.200
Wikidata	FISHDBC	-1.068	-1.255	-0.830	4.000
Wikidata*	Empty	-1.272	-1.351	-1.185	0.000
Wikidata*	ESCROW-P	-0.839	-1.124	-0.334	4.400
Wikidata*	COBWEB	-1.083	-1.252	-0.959	1.400
Wikidata*	FISHDBC	-1.010	-1.082	-0.815	3.600

Table 52: Paired ESCROW advantage over each certified alternative. Intervals resample five order-level differences, not individual records or blocks. Positive orders count the observed directions; these corpora are not a representative population sample.

Corpus	Baseline	Mean gain	95% interval	Positive orders
Amazon	COBWEB	0.401	[0.296, 0.518]	5/5
Amazon	FISHDBC	0.729	[0.620, 0.853]	5/5
Amazon*	COBWEB	0.377	[0.164, 0.652]	5/5
Amazon*	FISHDBC	0.660	[0.496, 0.824]	5/5
Wikidata	COBWEB	0.525	[0.395, 0.698]	5/5
Wikidata	FISHDBC	0.223	[0.170, 0.297]	5/5
Wikidata*	COBWEB	0.244	[-0.034, 0.598]	4/5
Wikidata*	FISHDBC	0.172	[-0.124, 0.491]	3/5

E.8.1 Prediction under identical pruning decisions

The 42 reused inputs and all three selected graphs are exactly those of Appendix E.7. No generator or certification decision is rerun. Each graph is scored once with explicit nodes and once with the existing residual readout. At depths two through seven the mean pruned-minus-whole conditional score is -0.524 bits per record with explicit nodes: 15 of 18 streams lose. With residual readout, the mean is $+0.110$ and 13 of 18 streams improve. These signs are not the same as the ordinary mixture predictor's signs. Recovery does not change merely because the predictor changes. Its deeper losses remain a more stable finding than a general claim that pruning harms prediction.

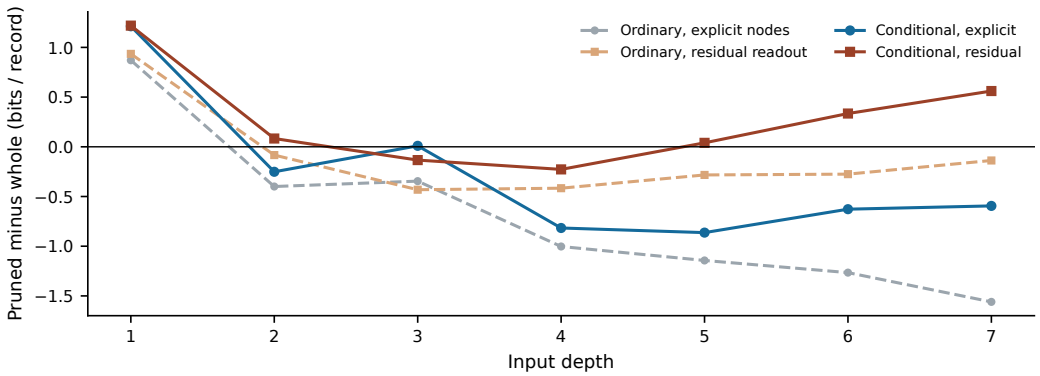


Figure 16: The same fixed graphs, two predictors and two group readouts. Three streams per depth. The ordinary and conditional analyses have separate frozen protocols and independent scoring audits. All axes use bits per record, but ordinary and conditional references differ; these curves compare paired pruning changes, not raw performance between the two scoring systems.

Table 53: Conditional predictor with semantic sharing on fixed graphs. Mean pruning change and count of positive changes out of three streams. Readout is fixed within each comparison.

Depth	Explicit change	Positive	Residual change	Positive
1	1.213	3/3	1.219	3/3
2	-0.251	0/3	0.083	3/3
3	0.010	1/3	-0.134	1/3
4	-0.817	0/3	-0.227	1/3
5	-0.863	1/3	0.040	2/3
6	-0.628	1/3	0.335	3/3
7	-0.595	0/3	0.561	3/3

E.8.2 CANDIDATE SUPPLY AT FIXED DEPTH AND FIELD COUNT

We vary FISHDBC minimum cluster size over 2, 5, and 10, keeping minimum samples at 5. The test uses the three fresh seeds at depths three and seven. Each candidate is evaluated native, whole-certified, and pruned-certified, with both readouts. This changes candidate composition as well as count, so it is not a pure intervention on the number of groups. No setting is selected from these outcomes.

At depth three, mean native group counts are 52.67, 18.00, and 10.67. Pruning returns six groups at each setting. Its mean membership-F1 changes relative to the whole arm are $+0.081$, -0.050, and -0.064. Candidate supply can therefore change the direction of the pruning comparison in this condition. At depth seven the corresponding changes are -0.128, -0.089, and -0.067. Even the largest native pool, 64.67 groups on average, is smaller than the reference hierarchy. Thus this test does not establish what pruning would do to a depth-seven pool with more groups than the reference, nor whether those extra groups would be useful. Group counts alone are not a mechanism.

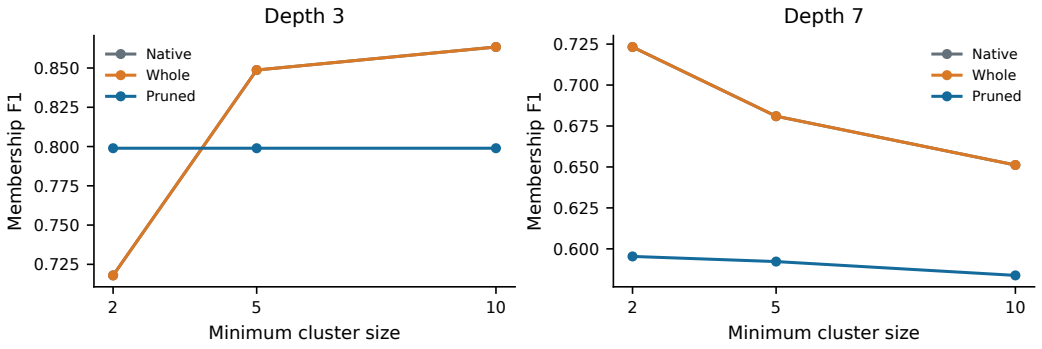


Figure 17: Fresh candidate-supply test. Three streams per point. Whole-certificate and native curves overlap because every signal candidate is accepted without pruning. Depth and field count stay fixed within each panel.

Table 54: Supply-test means. m is minimum cluster size. Score change is the conditional pruned-minus-whole rate with explicit nodes, averaged over three future blocks.

Depth	m	Native groups	Pruned groups	Native F1	Whole F1	Pruned F1	Score change
3	2	52.667	6.000	0.718	0.718	0.799	0.117
3	5	18.000	6.000	0.849	0.849	0.799	-0.073
3	10	10.667	6.000	0.863	0.863	0.799	-0.217
7	2	64.667	16.000	0.723	0.723	0.595	-0.306
7	5	30.000	15.333	0.681	0.681	0.592	-1.285
7	10	22.000	14.333	0.651	0.651	0.584	-1.837

E.8.3 INDEPENDENT SCORING AUDIT AND REPRODUCIBILITY

A separate probability-space implementation computes expert updates directly and sums cell log probability ratios against the remaining-count reference. All 252 diagnostic scores agree within 9.08×10^{-8} bits per record. Shared-count vectors use float32 arithmetic under NumPy 2. With the same vector values promoted to float64 only for the audit, the maximum discrepancy is 2.52×10^{-11} bits per block. Experimental code and outputs are unchanged.

For a small two-column type, enumeration of all nine arrangements gives reference probability sum one. The production sum is 0.9999999893; this is a recorded numerical approximation, not exact floating-point normalization. The theorem concerns ideal probabilities. The audit shares the prefix-derived kernels and does not independently validate their feature construction. Finite checks cannot prove all possible inputs.

The protocol records Python, NumPy, SciPy, COBWEB and FISHDBC package versions. Two local worker processes run with BLAS threads limited to one. A predeclared 600-second limit applies to each case. All 175 declared cases completed without failure, yielding 820 graph/readout outcomes and 1,956 block scores. The verification reconstructs all 381 certificates with zero discrepancy. All three certified proposers select zero groups on all 44 matched benchmark null cases. The package contains every frozen input, candidate and selected graph, matching, certificate, block score, and relevant source hash. Run `scripts/current_benchmark.py --run` to complete missing cases; `--freeze` must not overwrite the existing protocol. To rerun completed cases, use a separate copy and move that copy's job directory aside. Run `scripts/audit_current_predictor.py` for the independent scoring diagnostic and `scripts/summarize_current.py` to rebuild figures and tables.

E.8.4 Complete case summaries

The following tables print every declared case's score means and selected group counts, along with synthetic membership recovery. The accompanying JSON retains every individual block value, parent and ancestor metric, graph, and failed case, if any. Means do not discard empty outputs or unfavorable blocks.

Table 55: Every current-benchmark case: mean conditional bits per record over three later blocks. N marks a matched null, L a ladder case, D a depth case; A and W denote Amazon and Wikidata, with * marking removed fields. The final integer identifies the declared seed or order. Every block score is retained in the accompanying JSON. Rows 1 to 30.

Case	Empty	ESCROW	COBWEB	FISHDBC
D1/1	-0.094	-0.094	1.739	1.657
D1/1N	-0.103	-0.103	-0.103	-0.103
D1/2	-0.102	1.258	1.706	1.706
D1/2N	-0.074	-0.074	-0.074	-0.074
D1/3	-0.099	-0.099	1.697	1.697
D1/3N	-0.075	-0.075	-0.075	-0.075
D2/1	-0.097	5.698	5.520	5.176
D2/1N	-0.092	-0.092	-0.092	-0.092
D2/2	-0.130	5.314	5.450	4.518
D2/2N	-0.126	-0.126	-0.126	-0.126
D2/3	-0.184	5.577	5.498	4.806
D2/3N	-0.076	-0.076	-0.076	-0.076
D3/1	-0.119	12.093	10.896	10.418
D3/1N	-0.151	-0.151	-0.151	-0.151
D3/2	-0.143	11.122	10.542	10.122
D3/2N	-0.096	-0.096	-0.096	-0.096
D3/3	-0.091	11.306	10.603	9.991
D3/3N	-0.123	-0.123	-0.123	-0.123
D4/1	-0.164	18.536	18.059	16.234
D4/1N	-0.171	-0.171	-0.171	-0.171
D4/2	-0.139	18.972	18.418	16.183
D4/2N	-0.147	-0.147	-0.147	-0.147
D4/3	-0.084	18.700	18.316	16.696
D4/3N	-0.123	-0.123	-0.123	-0.123
D5/1	-0.136	24.636	24.779	22.703
D5/1N	-0.216	-0.216	-0.216	-0.216
D5/2	-0.189	24.961	24.603	22.199
D5/2N	-0.144	-0.144	-0.144	-0.144
D5/3	-0.226	25.456	25.042	23.217
D5/3N	-0.154	-0.154	-0.154	-0.154

Table 56: Every current-benchmark case: mean conditional bits per record over three later blocks. N marks a matched null, L a ladder case, D a depth case; A and W denote Amazon and Wikidata, with * marking removed fields. The final integer identifies the declared seed or order. Every block score is retained in the accompanying JSON. Rows 31 to 60.

Case	Empty	ESCROW	COBWEB	FISHDBC
D6/1	-0.249	29.793	29.533	25.228
D6/1N	-0.221	-0.221	-0.221	-0.221
D6/2	-0.254	29.255	29.495	27.492
D6/2N	-0.200	-0.200	-0.200	-0.200
D6/3	-0.244	29.358	29.775	26.802
D6/3N	-0.269	-0.269	-0.269	-0.269
D7/1	-0.326	31.138	31.879	27.618
D7/1N	-0.347	-0.347	-0.347	-0.347
D7/2	-0.295	32.294	32.720	28.387
D7/2N	-0.339	-0.339	-0.339	-0.339
D7/3	-0.324	32.281	32.899	30.814
D7/3N	-0.268	-0.268	-0.268	-0.268
L0/1N	-0.158	-0.158	-0.158	-0.158
L0/2N	-0.149	-0.149	-0.149	-0.149
L0/3N	-0.157	-0.157	-0.157	-0.157
L0.05/1	-0.157	-0.157	-0.157	-0.157
L0.05/2	-0.142	-0.142	-0.142	-0.142
L0.05/3	-0.171	-0.171	-0.171	-0.171
L0.1/1	-0.154	-0.154	-0.154	-0.154
L0.1/2	-0.140	-0.140	-0.140	-0.140
L0.1/3	-0.175	-0.175	-0.175	-0.175
L0.2/1	-0.141	-0.141	-0.141	-0.141
L0.2/2	-0.158	-0.158	-0.158	-0.158
L0.2/3	-0.181	-0.181	-0.181	-0.181
L0.3/1	-0.145	-0.145	-0.145	-0.145
L0.3/2	-0.165	-0.165	-0.165	-0.165
L0.3/3	-0.177	-0.177	-0.177	-0.177
L0.4/1	-0.155	-0.155	-0.155	-0.155
L0.4/2	-0.163	-0.163	-0.163	-0.163
L0.4/3	-0.171	-0.171	-0.171	-0.171

Table 57: Every current-benchmark case: mean conditional bits per record over three later blocks. N marks a matched null, L a ladder case, D a depth case; A and W denote Amazon and Wikidata, with * marking removed fields. The final integer identifies the declared seed or order. Every block score is retained in the accompanying JSON. Rows 61 to 90.

Case	Empty	ESCROW	COBWEB	FISHDBC
L0.5/1	-0.158	-0.158	-0.158	-0.158
L0.5/2	-0.157	-0.157	-0.157	-0.157
L0.5/3	-0.177	-0.177	-0.177	-0.177
L0.65/1	-0.149	-0.149	-0.149	-0.149
L0.65/2	-0.178	-0.178	-0.178	-0.178
L0.65/3	-0.173	-0.173	-0.173	-0.173
L0.8/1	-0.135	-0.135	-0.135	-0.135
L0.8/2	-0.189	-0.189	-0.189	-0.189
L0.8/3	-0.159	-0.159	-0.159	-0.159
L0.9/1	-0.132	-0.132	-0.132	-0.132
L0.9/2	-0.185	-0.185	-0.185	-0.185
L0.9/3	-0.165	-0.165	-0.165	-0.165
L1/1	-0.147	-0.147	7.621	-0.147
L1/2	-0.191	0.658	8.279	-0.191
L1/3	-0.174	0.689	-0.174	-0.174
A1	-0.237	1.542	1.038	0.603
A1N	-0.266	-0.266	-0.266	-0.266
A2	-0.265	1.642	1.332	0.936
A2N	-0.241	-0.241	-0.241	-0.241
A3	-0.274	1.404	1.143	0.854
A3N	-0.291	-0.291	-0.291	-0.291
A4	-0.282	1.126	0.786	0.462
A4N	-0.281	-0.281	-0.281	-0.281
A5	-0.261	1.456	0.864	0.669
A5N	-0.294	-0.294	-0.294	-0.294
A*1	-0.216	0.653	0.457	0.256
A*1N	-0.253	-0.253	-0.253	-0.253
A*2	-0.240	1.049	0.875	0.224
A*2N	-0.245	-0.245	-0.245	-0.245
A*3	-0.251	0.822	0.683	-0.048

Table 58: Every current-benchmark case: mean conditional bits per record over three later blocks. N marks a matched null, L a ladder case, D a depth case; A and W denote Amazon and Wikidata, with * marking removed fields. The final integer identifies the declared seed or order. Every block score is retained in the accompanying JSON. Rows 91 to 115.

Case	Empty	ESCROW	COBWEB	FISHDBC
A*3N	-0.255	-0.255	-0.255	-0.255
A*4	-0.263	0.592	-0.263	-0.139
A*4N	-0.261	-0.261	-0.261	-0.261
A*5	-0.236	0.649	0.126	0.173
A*5N	-0.235	-0.235	-0.235	-0.235
W1	-1.560	-0.904	-1.314	-1.255
W1N	-1.500	-1.500	-1.500	-1.500
W2	-1.515	-0.588	-1.457	-0.830
W2N	-1.416	-1.416	-1.416	-1.416
W3	-1.420	-0.977	-1.319	-1.116
W3N	-1.516	-1.516	-1.516	-1.516
W4	-1.604	-1.030	-1.505	-1.222
W4N	-1.595	-1.595	-1.595	-1.595
W5	-1.536	-0.727	-1.258	-0.919
W5N	-1.584	-1.584	-1.584	-1.584
W*1	-1.296	-0.953	-0.988	-1.073
W*1N	-1.233	-1.233	-1.233	-1.233
W*2	-1.252	-0.334	-1.252	-1.072
W*2N	-1.224	-1.224	-1.224	-1.224
W*3	-1.185	-0.755	-0.959	-1.082
W*3N	-1.182	-1.182	-1.182	-1.182
W*4	-1.351	-1.028	-1.228	-1.010
W*4N	-1.256	-1.256	-1.256	-1.256
W*5	-1.276	-1.124	-0.986	-0.815
W*5N	-1.293	-1.293	-1.293	-1.293

Table 59: Every selected group count and synthetic membership F1. E, C and F denote ESCROW, COBWEB and FISHDBC. Real reference recovery is not a declared outcome; null F1 is not interpreted. Parent and ancestor metrics are retained in JSON. Rows 1 to 30.

Case	E groups	E F1	C groups	C F1	F groups	F F1
D1/1	0	0.000	2	1.000	2	0.998
D1/1N	0	n/a	0	n/a	0	n/a
D1/2	2	0.768	2	1.000	2	1.000
D1/2N	0	n/a	0	n/a	0	n/a
D1/3	0	0.000	2	1.000	2	1.000
D1/3N	0	n/a	0	n/a	0	n/a
D2/1	7	0.597	6	1.000	5	0.923
D2/1N	0	n/a	0	n/a	0	n/a
D2/2	7	0.572	6	1.000	3	0.772
D2/2N	0	n/a	0	n/a	0	n/a
D2/3	5	0.575	8	0.934	4	0.822
D2/3N	0	n/a	0	n/a	0	n/a
D3/1	9	0.484	6	0.689	6	0.799
D3/1N	0	n/a	0	n/a	0	n/a
D3/2	7	0.472	9	0.885	6	0.798
D3/2N	0	n/a	0	n/a	0	n/a
D3/3	6	0.469	11	0.902	6	0.799
D3/3N	0	n/a	0	n/a	0	n/a
D4/1	11	0.416	12	0.734	9	0.744
D4/1N	0	n/a	0	n/a	0	n/a
D4/2	10	0.386	14	0.774	10	0.769
D4/2N	0	n/a	0	n/a	0	n/a
D4/3	10	0.385	12	0.711	10	0.765
D4/3N	0	n/a	0	n/a	0	n/a
D5/1	16	0.313	17	0.648	13	0.716
D5/1N	0	n/a	0	n/a	0	n/a
D5/2	15	0.323	17	0.717	12	0.702
D5/2N	0	n/a	0	n/a	0	n/a
D5/3	14	0.324	20	0.662	13	0.725
D5/3N	0	n/a	0	n/a	0	n/a

Table 60: Every selected group count and synthetic membership F1. E, C and F denote ESCROW, COBWEB and FISHDBC. Real reference recovery is not a declared outcome; null F1 is not interpreted. Parent and ancestor metrics are retained in JSON. Rows 31 to 60.

Case	E groups	E F1	C groups	C F1	F groups	F F1
D6/1	19	0.269	25	0.638	14	0.615
D6/1N	0	n/a	0	n/a	0	n/a
D6/2	20	0.272	26	0.649	14	0.643
D6/2N	0	n/a	0	n/a	0	n/a
D6/3	17	0.279	25	0.618	15	0.666
D6/3N	0	n/a	0	n/a	0	n/a
D7/1	27	0.232	25	0.555	15	0.582
D7/1N	0	n/a	0	n/a	0	n/a
D7/2	23	0.240	24	0.552	13	0.572
D7/2N	0	n/a	0	n/a	0	n/a
D7/3	22	0.248	20	0.465	18	0.622
D7/3N	0	n/a	0	n/a	0	n/a
L0/1N	0	n/a	0	n/a	0	n/a
L0/2N	0	n/a	0	n/a	0	n/a
L0/3N	0	n/a	0	n/a	0	n/a
L0.05/1	0	0.000	0	0.000	0	0.000
L0.05/2	0	0.000	0	0.000	0	0.000
L0.05/3	0	0.000	0	0.000	0	0.000
L0.1/1	0	0.000	0	0.000	0	0.000
L0.1/2	0	0.000	0	0.000	0	0.000
L0.1/3	0	0.000	0	0.000	0	0.000
L0.2/1	0	0.000	0	0.000	0	0.000
L0.2/2	0	0.000	0	0.000	0	0.000
L0.2/3	0	0.000	0	0.000	0	0.000
L0.3/1	0	0.000	0	0.000	0	0.000
L0.3/2	0	0.000	0	0.000	0	0.000
L0.3/3	0	0.000	0	0.000	0	0.000
L0.4/1	0	0.000	0	0.000	0	0.000
L0.4/2	0	0.000	0	0.000	0	0.000
L0.4/3	0	0.000	0	0.000	0	0.000

Table 61: Every selected group count and synthetic membership F1. E, C and F denote ESCROW, COBWEB and FISHDBC. Real reference recovery is not a declared outcome; null F1 is not interpreted. Parent and ancestor metrics are retained in JSON. Rows 61 to 90.

Case	E groups	E F1	C groups	C F1	F groups	F F1
L0.5/1	0	0.000	0	0.000	0	0.000
L0.5/2	0	0.000	0	0.000	0	0.000
L0.5/3	0	0.000	0	0.000	0	0.000
L0.65/1	0	0.000	0	0.000	0	0.000
L0.65/2	0	0.000	0	0.000	0	0.000
L0.65/3	0	0.000	0	0.000	0	0.000
L0.8/1	0	0.000	0	0.000	0	0.000
L0.8/2	0	0.000	0	0.000	0	0.000
L0.8/3	0	0.000	0	0.000	0	0.000
L0.9/1	0	0.000	0	0.000	0	0.000
L0.9/2	0	0.000	0	0.000	0	0.000
L0.9/3	0	0.000	0	0.000	0	0.000
L1/1	0	0.000	8	0.742	0	0.000
L1/2	1	0.171	9	0.893	0	0.000
L1/3	1	0.165	0	0.000	0	0.000
A1	9	n/a	14	n/a	4	n/a
A1N	0	n/a	0	n/a	0	n/a
A2	8	n/a	13	n/a	8	n/a
A2N	0	n/a	0	n/a	0	n/a
A3	5	n/a	16	n/a	6	n/a
A3N	0	n/a	0	n/a	0	n/a
A4	9	n/a	17	n/a	3	n/a
A4N	0	n/a	0	n/a	0	n/a
A5	8	n/a	10	n/a	6	n/a
A5N	0	n/a	0	n/a	0	n/a
A*1	9	n/a	11	n/a	5	n/a
A*1N	0	n/a	0	n/a	0	n/a
A*2	9	n/a	10	n/a	3	n/a
A*2N	0	n/a	0	n/a	0	n/a
A*3	8	n/a	16	n/a	3	n/a

Table 62: Every selected group count and synthetic membership F1. E, C and F denote ESCROW, COBWEB and FISHDBC. Real reference recovery is not a declared outcome; null F1 is not interpreted. Parent and ancestor metrics are retained in JSON. Rows 91 to 115.

Case	E groups	E F1	C groups	C F1	F groups	F F1
A*3N	0	n/a	0	n/a	0	n/a
A*4	10	n/a	0	n/a	2	n/a
A*4N	0	n/a	0	n/a	0	n/a
A*5	6	n/a	10	n/a	5	n/a
A*5N	0	n/a	0	n/a	0	n/a
W1	4	n/a	1	n/a	3	n/a
W1N	0	n/a	0	n/a	0	n/a
W2	3	n/a	2	n/a	5	n/a
W2N	0	n/a	0	n/a	0	n/a
W3	8	n/a	4	n/a	4	n/a
W3N	0	n/a	0	n/a	0	n/a
W4	11	n/a	2	n/a	5	n/a
W4N	0	n/a	0	n/a	0	n/a
W5	8	n/a	2	n/a	3	n/a
W5N	0	n/a	0	n/a	0	n/a
W*1	3	n/a	2	n/a	1	n/a
W*1N	0	n/a	0	n/a	0	n/a
W*2	6	n/a	0	n/a	3	n/a
W*2N	0	n/a	0	n/a	0	n/a
W*3	6	n/a	2	n/a	4	n/a
W*3N	0	n/a	0	n/a	0	n/a
W*4	5	n/a	2	n/a	6	n/a
W*4N	0	n/a	0	n/a	0	n/a
W*5	2	n/a	1	n/a	4	n/a
W*5N	0	n/a	0	n/a	0	n/a

Table 63: Every current-predictor diagnostic score, conditional bits per record on the fixed 100-record block. R denotes residual readout. These are the reused cases, not fresh confirmation streams. Rows 1 to 24.

Case	Native	Whole	Pruned	Native R	Whole R	Pruned R
D1/1	1.585	-0.101	1.701	1.585	-0.101	1.701
D1/1N	-0.086	-0.079	-0.079	-0.086	-0.079	-0.079
D2/1	5.338	5.338	5.237	5.338	5.295	5.400
D2/1N	-0.098	-0.074	-0.074	-0.098	-0.074	-0.074
D3/1	10.327	10.327	10.498	10.327	10.375	10.498
D3/1N	-0.132	-0.132	-0.132	-0.132	-0.132	-0.132
D4/1	16.673	16.673	16.197	16.673	17.105	16.828
D4/1N	-0.184	-0.176	-0.176	-0.184	-0.176	-0.176
D5/1	23.169	23.169	21.931	23.169	23.704	23.845
D5/1N	-0.284	-0.237	-0.237	-0.284	-0.237	-0.237
D6/1	27.587	27.587	27.987	27.587	28.025	28.671
D6/1N	-0.327	-0.270	-0.270	-0.327	-0.270	-0.270
D7/1	29.645	29.645	29.090	29.645	30.625	30.995
D7/1N	-0.278	-0.235	-0.235	-0.278	-0.235	-0.235
D1/2	1.643	1.643	1.699	1.643	1.624	1.699
D1/2N	-0.110	-0.096	-0.096	-0.110	-0.096	-0.096
D2/2	5.301	5.301	4.863	5.301	5.291	5.368
D2/2N	-0.100	-0.083	-0.083	-0.100	-0.083	-0.083
D3/2	10.727	10.727	10.595	10.727	11.045	10.595
D3/2N	-0.132	-0.124	-0.124	-0.132	-0.124	-0.124
D4/2	17.170	17.170	15.837	17.170	17.811	17.376
D4/2N	-0.159	-0.157	-0.157	-0.159	-0.157	-0.157
D5/2	23.297	23.297	21.748	23.297	23.749	23.643
D5/2N	-0.167	-0.190	-0.190	-0.167	-0.190	-0.190

Table 64: Every current-predictor diagnostic score, conditional bits per record on the fixed 100-record block. R denotes residual readout. These are the reused cases, not fresh confirmation streams. Rows 25 to 42.

Case	Native	Whole	Pruned	Native R	Whole R	Pruned R
D6/2	27.555	27.555	26.300	27.555	28.452	28.593
D6/2N	-0.244	-0.208	-0.208	-0.244	-0.208	-0.208
D7/2	30.415	30.415	29.420	30.415	31.131	31.552
D7/2N	-0.270	-0.312	-0.312	-0.270	-0.312	-0.312
D1/3	1.673	-0.056	1.723	1.673	-0.056	1.723
D1/3N	-0.113	-0.125	-0.125	-0.113	-0.125	-0.125
D2/3	5.190	5.190	4.974	5.190	5.287	5.356
D2/3N	-0.147	-0.138	-0.138	-0.147	-0.138	-0.138
D3/3	10.351	10.351	10.342	10.351	10.416	10.342
D3/3N	-0.152	-0.114	-0.114	-0.152	-0.114	-0.114
D4/3	17.455	17.455	16.814	17.455	17.817	17.846
D4/3N	-0.184	-0.183	-0.183	-0.184	-0.183	-0.183
D5/3	24.045	24.045	24.243	24.045	24.326	24.410
D5/3N	-0.199	-0.144	-0.144	-0.199	-0.144	-0.144
D6/3	27.521	27.521	26.492	27.521	28.229	28.446
D6/3N	-0.305	-0.233	-0.233	-0.305	-0.233	-0.233
D7/3	30.161	30.161	29.927	30.161	30.823	31.714
D7/3N	-0.431	-0.360	-0.360	-0.431	-0.360	-0.360

Table 65: Every candidate-supply case. Change is pruned minus whole conditional bits per record, averaged over three later blocks. R denotes residual readout. The minimum cluster size is m; minimum samples stays five. Rows 1 to 18.

Case	Native groups	Pruned groups	Whole F1	Pruned F1	Change	Change R
D3/m2/1	56	6	0.696	0.799	0.226	0.110
D3/m5/1	14	6	0.815	0.799	0.128	0.049
D3/m10/1	8	6	0.818	0.799	0.008	-0.089
D3/m2/2	48	6	0.753	0.798	-0.070	-0.121
D3/m5/2	20	6	0.869	0.798	-0.237	-0.289
D3/m10/2	12	6	0.891	0.798	-0.353	-0.387
D3/m2/3	54	6	0.706	0.799	0.196	0.085
D3/m5/3	20	6	0.862	0.799	-0.111	-0.251
D3/m10/3	12	6	0.881	0.799	-0.306	-0.450
D7/m2/1	70	15	0.725	0.582	-0.978	0.714
D7/m5/1	28	15	0.670	0.582	-2.053	-0.118
D7/m10/1	22	13	0.646	0.565	-2.590	-0.298
D7/m2/2	66	13	0.717	0.572	-0.575	1.353
D7/m5/2	32	13	0.683	0.572	-1.875	0.182
D7/m10/2	22	12	0.648	0.564	-2.258	0.146
D7/m2/3	58	20	0.729	0.632	0.636	1.326
D7/m5/3	30	18	0.690	0.622	0.071	0.767
D7/m10/3	22	18	0.660	0.622	-0.662	0.104

E.9 SEPARATING RECURSIVE RECOVERY FROM PARENT-BASED INSERTION

Question and design. Do the two proposal mechanisms recover different structures on the same records, and could graph export or the evaluator explain that difference? We use all 42 depth inputs from the matched development study: depths one through seven, seeds 34000, 34001, and 34002, with signal/null pairs and 300 records per case. This is a diagnostic on reused development inputs, not another independent confirmation set. Its protocol, source hashes, and input hashes are supplied in `evidence/depth_code_audit/`.

For each input, we decode the saved recursive graph from its EHR05 bitstream, check that it re-encodes byte for byte, and match its records, memberships, field supports, parent links, and recovery scores to the saved result. We then run parent-based insertion on exactly those records in the same order. Both graphs are evaluated native, with whole-graph conditional certification, and with conditional pruning. All arms use the same reference memberships and evaluator. The conditional arms use `FastCertifier`, `values=min`, and margin 10. This is a common postprocessing comparison; it does not redefine the native recursive engine's EHR05 selection rule.

Code paths. The recursive engine is `FastCachedBoundaryCascadeGraph`, using recursive workers, repair cadence 100, candidate budget 4,096, maximum components 16, and a terminal repair. The alternative is `ParentCentred` with its default insertion rule. Its insertion method does not call the separately defined merge routine or the recursive repair engine. It scores all fields with its own smoothed counts. Thus the two methods differ in search operations and candidate supply, not just an argument value or random seed. Neither receives the reference hierarchy during construction.

Table 66: Mean membership F1 on identical signal inputs, three seeds per depth. C is recursive cascade; P is parent-based insertion. Whole and pruned use the same conditional certifier, without and with node deletion respectively. Native C uses its own complete-code search gate. Scores are rounded; depth-two C is 0.99972, not exactly one on every stream.

Depth	C raw	C whole	C pruned	P raw	P whole	P pruned
1	0.8046	0.8046	0.8046	0.2792	0.0000	0.0000
2	0.9997	0.9997	0.9997	0.4856	0.0000	0.6122
3	0.8773	0.8773	0.8667	0.4453	0.1484	0.4795
4	0.7884	0.7884	0.7884	0.4039	0.4039	0.3740
5	0.7058	0.7058	0.7058	0.4079	0.4079	0.3191
6	0.5712	0.5712	0.5659	0.3896	0.3896	0.2748
7	0.5526	0.5526	0.5526	0.3393	0.3393	0.2383

Table 67: Mean returned depth for the same six arms. The parent-based pool is already shallow before certification; subsequent pruning often leaves a flat selected graph.

Depth	C raw	C whole	C pruned	P raw	P whole	P pruned
1	1.6667	1.6667	1.6667	2.3333	0.0000	0.0000
2	2.0000	2.0000	2.0000	2.6667	0.0000	1.0000
3	2.6667	2.6667	2.6667	2.3333	0.6667	1.0000
4	3.0000	3.0000	3.0000	2.3333	2.3333	1.0000
5	3.0000	3.0000	3.0000	2.0000	2.0000	1.0000
6	2.6667	2.6667	2.6667	2.0000	2.0000	1.0000
7	3.0000	3.0000	3.0000	2.0000	2.0000	1.3333

What changes and what survives. The recursive output exceeds parent-based output after pruning on all 21 signal streams. At depth two the mean membership F1 values are 0.99972 and 0.61217; at depth seven they are 0.55260 and 0.23827. Whole-graph conditional certification accepts all 21 recursive signal outputs and preserves their memberships, links, and recovery. All 21 recursive null outputs remain empty. Pruning alters two recursive outputs, one at depth three and one at depth six, with small recovery losses. It is therefore incorrect to infer a general recursive-recovery collapse from the parent-based arm.

This distinction does not establish that recursive search wins every comparison. Native FISHDBC has higher mean membership F1 at depths four through seven on these inputs. The cascade's strong shallow results and its remaining deep-recovery gap are both retained in Figure 3. The three fresh-seed parent-based benchmark in Appendix E.8 remains valid for that explicitly named implementation.

Cold replay and graph export. Two runs start from empty state and repeat the original recursive computation at depths two and seven with seed 34000. Both reproduce the saved memberships, supports, parent links, and all three recovery F1 scores exactly. The depth-two replay has F1 1.0 for membership, parent links, and ancestor links. This is a correctness probe, not a new estimate of average performance.

We separately inspect the internal parent-based trees on those two inputs. They have 300 nonroot nodes and maximum internal depth three in each case. After the declared minimum-size filter, they export 96 and 81 groups, both of depth two. Every eligible unique group and every minimal-inclusion parent link is retained. There is no missing exported level beyond the stated removal of singleton groups and duplicate memberships. The procedure genuinely supplies a different, shallow candidate pool. This inspection identifies a search difference; it does not isolate one nuisance field as its cause.

Independent metric checks. Every arm's membership and direct-link counts are recomputed independently. The identity check returns F1 one on supplied reference graphs. Computing pairwise group F1 directly as $2|A \cap B|/(|A| + |B|)$ can select a different optimal assignment from the precision/recall form because of floating-point ties. This occurs for parent-native and parent-whole arms in two depth-six cases. Both assignments have the same summed group-F1 objective within 10^{-9}. We retain the primary matching unchanged and save both metric outcomes. Neither alternative reverses the recursive-versus-parent result. This separates arithmetic verification from ambiguity in the matching convention.

Integrity and diagnostic amendments. All 42 cases and 252 arm outcomes are retained. All 168 conditional certificates reconstruct from the saved pre-threshold states. Two diagnostic amendments are recorded: serialization of set-valued deletion logs, and explicit handling of equal-objective matching alternatives after two assertions exposed them. Initial failure logs and outcomes are preserved. Neither amendment changes a proposer, production metric, input, selected graph, or declared comparison. The 175-case parent-based study remains frozen and unchanged.

Run `scripts/depth_code_audit.py --run paired` to complete missing paired cases, and use `--run replay` for the two cold replays. Existing outcomes are cached; recompute only in a separate copy. Run `scripts/audit_parent_export.py` for the export and identity checks, and `scripts/summarize_depth_audit.py` to verify certificates and rebuild this figure and these tables. The source package retains all graphs, owners, mappings, and recovery measures.

Table 68: Every paired diagnostic, cases 1–21. Signal cells report membership F1; cases ending N are nulls and report group counts. C and P have the meanings above. Case d2/0 is depth two, seed 34000; the final digit identifies the seed.

Case	C raw	C whole	C pruned	P raw	P whole	P pruned
d1/0	0.8253	0.8253	0.8253	0.2923	0.0000	0.0000
d1/0N	0	0	0	88	0	0
d1/1	0.7800	0.7800	0.7800	0.3311	0.0000	0.0000
d1/1N	0	0	0	78	0	0
d1/2	0.8086	0.8086	0.8086	0.2142	0.0000	0.0000
d1/2N	0	0	0	83	0	0
d2/0	1.0000	1.0000	1.0000	0.4835	0.0000	0.6127
d2/0N	0	0	0	87	0	0
d2/1	0.9992	0.9992	0.9992	0.4947	0.0000	0.6159
d2/1N	0	0	0	87	0	0
d2/2	1.0000	1.0000	1.0000	0.4786	0.0000	0.6079
d2/2N	0	0	0	72	0	0
d3/0	0.9104	0.9104	0.9104	0.4497	0.0000	0.4759
d3/0N	0	0	0	92	0	0
d3/1	0.8000	0.8000	0.8000	0.4451	0.4451	0.4789
d3/1N	0	0	0	83	0	0
d3/2	0.9215	0.9215	0.8896	0.4412	0.0000	0.4836
d3/2N	0	0	0	79	0	0
d4/0	0.8195	0.8195	0.8195	0.3949	0.3949	0.3536
d4/0N	0	0	0	104	0	0
d4/1	0.8183	0.8183	0.8183	0.4153	0.4153	0.3860

Table 69: Every paired diagnostic, cases 22–42, with the same conventions. No empty or unfavorable result is omitted. Complete relationship metrics are in the JSON.

Case	C raw	C whole	C pruned	P raw	P whole	P pruned
d4/1N	0	0	0	95	0	0
d4/2	0.7275	0.7275	0.7275	0.4017	0.4017	0.3825
d4/2N	0	0	0	103	0	0
d5/0	0.6175	0.6175	0.6175	0.4077	0.4077	0.3127
d5/0N	0	0	0	110	0	0
d5/1	0.7500	0.7500	0.7500	0.3918	0.3918	0.3216
d5/1N	0	0	0	108	0	0
d5/2	0.7500	0.7500	0.7500	0.4242	0.4242	0.3229
d5/2N	0	0	0	104	0	0
d6/0	0.6252	0.6252	0.6095	0.3781	0.3781	0.2800
d6/0N	0	0	0	87	0	0
d6/1	0.5000	0.5000	0.5000	0.3854	0.3854	0.2752
d6/1N	0	0	0	90	0	0
d6/2	0.5882	0.5882	0.5882	0.4055	0.4055	0.2692
d6/2N	0	0	0	93	0	0
d7/0	0.5268	0.5268	0.5268	0.3296	0.3296	0.2301
d7/0N	0	0	0	57	0	0
d7/1	0.6000	0.6000	0.6000	0.3323	0.3323	0.2367
d7/1N	0	0	0	60	0	0
d7/2	0.5310	0.5310	0.5310	0.3561	0.3561	0.2480
d7/2N	0	0	0	69	0	0

F ISOLATING THE OWNERSHIP DECISION

Question. Overlapping groups may explain different cells of the same record. Does the complete code support such groups, and how much does the assignment of cells affect its score? This component study fixes candidate memberships. It measures the code and its routing rule, not a search algorithm's ability to discover the supplied groups.

Declared inputs. We use 300 records, eight fields, and two groups of 120 records each. Their intersection contains 0, 12, 36, or 60 records, corresponding to 0, 10, 30, or 50% of each group. For each cell of a covered record, the generator chooses one eligible group uniformly and then one of that group's two field values uniformly. Uncovered records use two background values. The value inventories are disjoint across the two groups and background. Five seeds, 82000–82004, give 20 signal inputs. Independently shuffling each column gives 20 paired controls with exactly the same column counts. Both candidate memberships remain fixed on each control. No parameters are selected from the results. The frozen protocol, inputs, and source hashes are packaged.

Two assignments, one complete code. For the routing comparison, both groups support every field. The current deterministic rule assigns a shared cell to the last eligible group in topological order. An existing refinement starts from that route and makes up to eight rounds of leave-one-out assignment updates among eligible groups. Per field, it retains the shorter of its refined and initial assignments under the search code. We then price both assignments with the same complete conditional code and flagged value code. The refinement changes no records, memberships, links, or supports. It is a component alternative; the matched benchmark continues to use its declared deterministic rule.

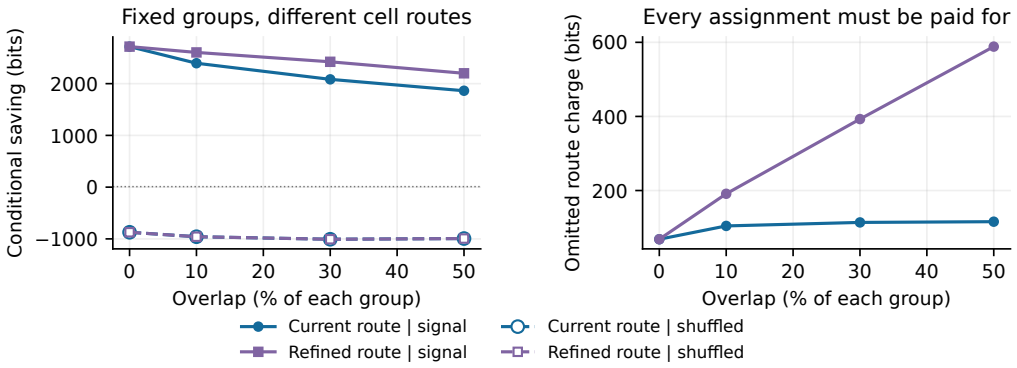


Figure 18: Ownership on fixed overlapping group pools. Each point averages five seeds. Left: complete conditional saving for signal and shuffled inputs. Right: the amount by which deleting the owner charge would inflate the signal score. That incomplete score is a diagnostic and has no certificate guarantee. The two shuffled curves coincide because refinement retains the current assignment on every control.

Outcomes. The unchanged selector, with its ordinary support choice and pruning, retains both supplied groups on all 20 signal inputs and selects no groups on all 20 controls. For the fixed-support routing comparison, refinement increases mean complete-code saving by 208.8, 339.2, and 337.1 bits at 10, 30, and 50% overlap. The disjoint case is unchanged. Better cell prediction can therefore pay for a more expensive route. Deleting the route charge inflates these signal scores by 68.5–588.4 bits across the condition means. On these controls, both incomplete scores still remain below the margin. A missing charge need not cause a false selection in every finite test.

F.1 EVERY OVERLAP OUTCOME

The tables separate fixed-support scores from the ordinary selector's output. All lengths are in bits. S uses every code term. "Route" is the included owner charge; deleting it would increase S by exactly that amount. "Kept" counts the explicit groups after ordinary support choice, pruning, and the 10-bit threshold. Seed indices 0–4 denote seeds 82000–82004.

Table 70: All 20 signal inputs with supplied overlapping candidate groups.

Overlap (%)	Seed	Current S	Refined S	Current route	Refined route	Kept
0	0	2715.7	2715.7	68.5	68.5	2
0	1	2715.7	2715.7	68.5	68.5	2
0	2	2715.7	2715.7	68.5	68.5	2
0	3	2715.6	2715.6	68.5	68.5	2
0	4	2715.7	2715.7	68.5	68.5	2
10	0	2414.9	2607.1	104.4	187.3	2
10	1	2418.0	2602.0	104.4	192.8	2
10	2	2369.1	2603.5	104.4	191.2	2
10	3	2363.3	2603.4	104.4	191.3	2
10	4	2408.0	2601.3	104.4	193.4	2
30	0	2093.3	2423.9	113.9	390.9	2
30	1	2071.9	2421.3	113.9	394.2	2
30	2	2082.0	2424.5	113.9	390.4	2
30	3	2094.6	2420.2	113.9	395.5	2
30	4	2074.0	2421.8	113.9	393.3	2
50	0	1840.6	2199.3	115.9	589.8	2
50	1	1858.0	2200.6	115.9	587.6	2
50	2	1885.9	2200.6	115.9	587.6	2
50	3	1866.9	2200.4	115.9	588.2	2
50	4	1863.7	2199.7	115.9	588.9	2

Table 71: All 20 paired shuffled controls, with the same fixed candidate memberships.

Overlap (%)	Seed	Current S	Refined S	Current route	Refined route	Kept
0	0	-883.4	-883.4	68.5	68.5	0
0	1	-872.2	-872.2	68.5	68.5	0
0	2	-868.7	-868.7	68.5	68.5	0
0	3	-869.5	-869.5	68.5	68.5	0
0	4	-866.5	-866.5	68.5	68.5	0
10	0	-954.1	-954.1	104.4	104.4	0
10	1	-954.1	-954.1	104.4	104.4	0
10	2	-953.1	-953.1	104.4	104.4	0
10	3	-964.7	-964.7	104.4	104.4	0
10	4	-964.8	-964.8	104.4	104.4	0
30	0	-1020.0	-1020.0	113.9	113.9	0
30	1	-1011.3	-1011.3	113.9	113.9	0
30	2	-989.0	-989.0	113.9	113.9	0
30	3	-1009.8	-1009.8	113.9	113.9	0
30	4	-1003.8	-1003.8	113.9	113.9	0
50	0	-983.2	-983.2	115.9	115.9	0
50	1	-999.6	-999.6	115.9	115.9	0
50	2	-1002.3	-1002.3	115.9	115.9	0
50	3	-990.8	-990.8	115.9	115.9	0
50	4	-990.9	-990.9	115.9	115.9	0

F.2 Exhaustive small cases and independent arithmetic

The small construction uses eight records, three fields, groups of four records, and intersection sizes zero through four. Seeds 81000–81002 and paired shuffling give 30 inputs and 90 field problems. Each cell can belong to background or an eligible group. Length separates by field: enumerating its legal routes gives the ownership minimum, with memberships, links, and supports fixed.

We enumerate 14,058 legal field routes. An independent implementation computes sequential owner, presence, and value probabilities and counts the remaining background arrangements. It agrees with the production array scorer within 5.12×10^{-13} bits. Every cell has exactly one eligible owner, including background, and every shuffled column preserves its counts.

Single-cell descent starts at the current route, takes the greatest strict reduction in the complete conditional length, and stops when no such move remains. The current route attains the exhaustive minimum in 0 of 90 fields; descent does so in 9 of 90. Their largest field gaps are 15.256 and 10.456 bits. Background alone is an optimal route in 87 of 90 fields. These forced-support examples are too small to justify the supplied groups: with its actual support selection and pruning enabled, the current certifier returns an empty graph on all 30 inputs. The routing gaps therefore measure a constrained component problem, not a recovery failure of the full selector.

Table 72: Every small input. Gaps sum the excess length above the exhaustive minimum over its three fields. Exact counts refer to those fields. Overlap is the number of shared records. Seed indices 0–2 denote seeds 81000–81002.

Overlap	Seed	Input	Current gap	Greedy gap	Current exact	Greedy exact
0	0	Signal	29.292	24.857	0/3	0/3
0	0	Shuffled	42.598	16.775	0/3	0/3
0	1	Signal	26.528	23.572	0/3	0/3
0	1	Shuffled	40.231	20.689	0/3	0/3
0	2	Signal	29.140	24.227	0/3	0/3
0	2	Shuffled	36.954	23.326	0/3	0/3
1	0	Signal	25.976	16.644	0/3	0/3
1	0	Shuffled	33.601	10.456	0/3	0/3
1	1	Signal	27.180	18.421	0/3	0/3
1	1	Shuffled	36.356	6.936	0/3	1/3
1	2	Signal	29.845	20.230	0/3	0/3
1	2	Shuffled	33.015	8.843	0/3	0/3
2	0	Signal	19.687	6.766	0/3	0/3
2	0	Shuffled	27.816	7.989	0/3	0/3
2	1	Signal	25.024	3.707	0/3	1/3
2	1	Shuffled	29.831	10.215	0/3	0/3
2	2	Signal	21.287	6.385	0/3	1/3
2	2	Shuffled	29.679	15.215	0/3	0/3
3	0	Signal	18.042	9.037	0/3	0/3
3	0	Shuffled	23.949	9.478	0/3	0/3
3	1	Signal	22.122	7.778	0/3	1/3
3	1	Shuffled	25.707	8.778	0/3	1/3
3	2	Signal	18.783	3.478	0/3	2/3
3	2	Shuffled	20.046	3.741	0/3	1/3
4	0	Signal	7.702	7.702	0/3	0/3
4	0	Shuffled	16.042	16.042	0/3	0/3
4	1	Signal	10.457	10.457	0/3	0/3
4	1	Shuffled	18.212	11.712	0/3	1/3
4	2	Signal	6.679	6.679	0/3	0/3
4	2	Shuffled	16.434	16.434	0/3	0/3

This study isolates accounting and assignment. It does not establish a unique true owner for observational data, prove that the current routing heuristic is optimal, or replace the matched recovery results. Deterministic routing that a decoder can reconstruct could be part of a different valid code; deleting the route term from this description is not a validation of such a replacement.

G PAIRED PREDICTION CHANGES AND EXPERT WEIGHTS

G.1 VARIATION ACROSS THE FIVE SAMPLED ORDERS

Figure 19 shows the effect of selection on the same input for each core proposer. Each point first averages three future blocks within one order. The figures do not treat those blocks as independent datasets. Squares are means; bars are descriptive 95% percentile intervals from all $5^5 = 3{,}125$ resamples of the five paired differences. Enumerating this distribution removes simulation noise. The tabulated 20,000-resample estimates use the same pairing and estimand. The head-effect bars in Figure 5 use the same exhaustive calculation.

These intervals describe variation in the sampled records, their order, and any proposer randomness under this fixed protocol. Five orders from the same corpus can share records. The intervals do not establish performance on unseen corpora or provide simultaneous coverage across every displayed comparison.

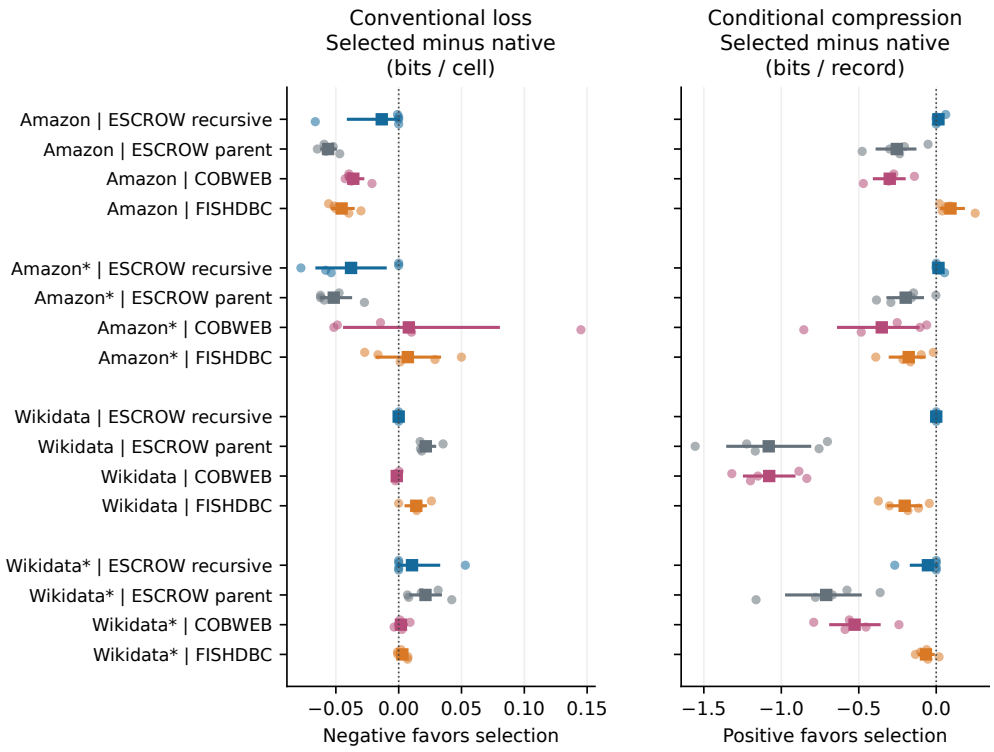


Figure 19: Selection changes conventional prediction and conditional compression differently. All five order-level differences appear for every proposer and variant. Negative conventional loss changes and positive conditional savings favor selection. Each comparison holds the prediction heads and future records fixed.

G.2 How the memory head changes its weights

We replay the first declared order, seed 72001, for both selected ESCROW proposers on all four real-data variants. The choice is fixed by order index, not performance. Figure 20 displays the conventional predictor's three expert weights, averaged across fields after each record. Fields have their own weights; the plotted mean is a summary, not the state used for prediction. The graph, prefix counts, and semantic similarities remain fixed across the three future blocks.

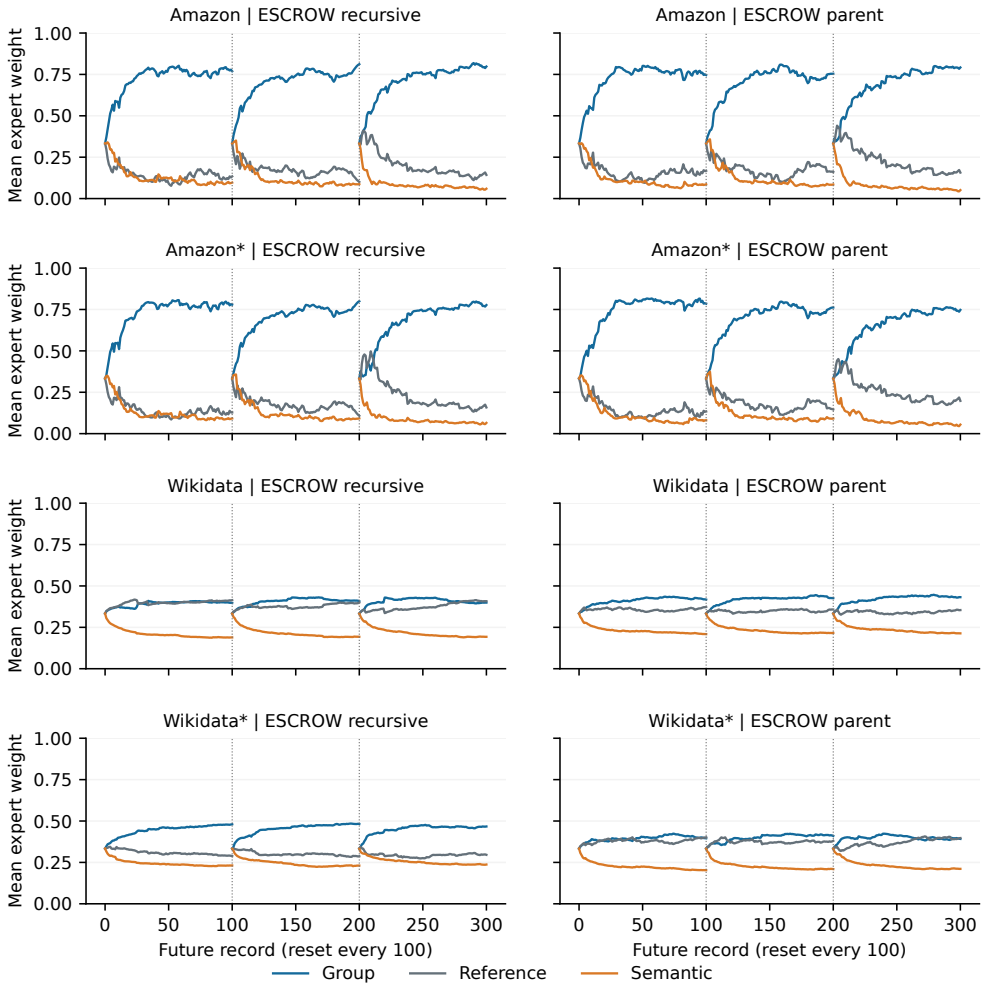


Figure 20: Group, reference, and semantic expert weights over the three future blocks. Each block starts at equal weights; dotted boundaries show the resets. Changes within a block reflect probability assigned to cells already observed. These traces illustrate adaptive weighting, not persistent memory or adaptation to changing graphs.

An independent replay of the probability update reproduces all 24 saved block losses within 3.64×10^{-12} bits. All expert weights stay positive and sum to one. The full per-record traces and the checked losses are supplied with the evidence.

H COMPLETE PROMPTS AND GENERATION CONTRACTS

This appendix prints the exact zero-shot and few-shot instructions, demonstrations, message builders, and hierarchy response contracts. Named slots stand for the records and group descriptions supplied at each call. The matched Wikidata study retains every record-filled request and raw response in the evidence package. The separate single-block study retains its builder and result summaries, but not all API traces. Hierarchy messages and schemas are copied byte for byte and hashed. Line wrapping below is typographical only.

H.1 REAL-DATA ZERO-SHOT PROMPT

There is no separately authored system message in the real-data runner. Qwen's tokenizer supplies its normal chat wrapper. Gemini receives the same single content string. Repeat the node and record lines as needed. When there are no nodes, the builder writes `(none yet)`.

```
Existing nodes (name: description):
- {NODE_NAME}: {NODE_DESCRIPTION}

New records:
[record {RECORD_ID}]
{KEY}={VALUE}

For every record above, decide which node it attaches to. Output a JSON array with
    one object per record, in record order:
- existing node: {"record": <id>, "node": "<existing node name, exactly as listed>"}
- new node: {"record": <id>, "node": "<new node name>", "new": true, "description":
    "<one line saying what kind of item the node stands for>"}
- background: {"record": <id>, "action": "background"}
A node created for an earlier record in this same answer may be reused by later
    records by name. Output the JSON array only.
```

H.2 REAL-DATA FEW-SHOT PROMPT, INCLUDING ALL THREE EXAMPLES

The complete few-shot prompt follows. Its final task instruction is the same zero-shot template. The demonstrations are unrelated bird records, not hidden examples from either evaluation corpus. They supply task guidance only to the few-shot LLM arm. ESCROW receives no such examples or domain instructions in any reported comparison; its counts and semantic similarities are learned from the arriving records.

```
Worked examples of the task, on an unrelated stream of bird observations. They show
    the
output format and the kind of judgement wanted. They say nothing about the records
    you will be given.

Example 1. No nodes exist yet.
Existing nodes (name: description):
(none yet)

New records:
[record 9000]
habitat=wetland
bill=long
diet=fish
[record 9001]
habitat=wetland
bill=long
diet=fish
[record 9002]
habitat=wetland
bill=long
diet=amphibian
[record 9003]
```

```
habitat=desert
bill=short
diet=seed

Answer:
[{"record": 9000, "node": "wading_fish_eater", "new": true, "description": "long
    billed wetland birds taking fish or amphibians"},
 {"record": 9001, "node": "wading_fish_eater"},
 {"record": 9002, "node": "wading_fish_eater"},
 {"record": 9003, "action": "background"}]

Records 9000 to 9002 agree on two fields and differ on one, so they are one node,
    and the node made for
record 9000 is reused by name later in the same answer. Record 9003 matches nothing
    else here, and one
record on its own is not evidence of a node, so it goes to background.

Example 2. Nodes exist.
Existing nodes (name: description):
- wading_fish_eater: long billed wetland birds taking fish or amphibians
- desert_seed_eater: short billed desert birds that eat seeds

New records:
[record 9004]
habitat=desert
bill=short
diet=seed
[record 9005]
habitat=forest
bill=hooked
diet=rodent
[record 9006]
habitat=forest
bill=hooked
diet=rodent

Answer:
[{"record": 9004, "node": "desert_seed_eater"},
 {"record": 9005, "node": "forest_raptor", "new": true, "description": "hooked bill
    forest birds that hunt rodents"},
 {"record": 9006, "node": "forest_raptor"}]

Record 9004 is attached to an existing node by its exact name. Records 9005 and 9006
     agree with each other
and with no existing node, so they open one.

Example 3. A near miss.
Existing nodes (name: description):
- forest_raptor: hooked bill forest birds that hunt rodents

New records:
[record 9007]
habitat=forest
bill=hooked
diet=bird

Answer:
[{"record": 9007, "node": "forest_raptor"}]

One field differs and the rest agree, so the record joins the node rather than
    starting another.

End of examples. The real records follow.
```

```
Existing nodes (name: description):
- {NODE_NAME}: {NODE_DESCRIPTION}

New records:
[record {RECORD_ID}]
{KEY}={VALUE}

For every record above, decide which node it attaches to. Output a JSON array with
    one object per record, in record order:
- existing node: {"record": <id>, "node": "<existing node name, exactly as listed>"}
- new node: {"record": <id>, "node": "<new node name>", "new": true, "description":
    "<one line saying what kind of item the node stands for>"}
- background: {"record": <id>, "action": "background"}
A node created for an earlier record in this same answer may be reused by later
    records by name. Output the JSON array only.
```

H.3 Exact construction of the real-data message

The function below is copied from the saved builder. Its `FEWSHOT` constant is exactly the demonstration block printed above, ending after "The real records follow." The code specifies record IDs, node order, field order, line breaks, and the no-node case. Dynamic slots in the printed templates expand through this function. Both real-data models use this text. In the matched Wikidata study, Qwen uses greedy decoding with at most 8,192 new tokens. Gemini uses temperature zero, low reasoning effort, and an 8,192-token output allowance. The separate single-block study uses limits of 1,400 and 4,096, respectively. The studies have separate result tables.

```
def make_prompt(nodes, chunk, start, fewshot=False):
    L = [FEWSHOT, ""] if fewshot else []
    L += ["Existing nodes (name: description):"]
    L += [f"- {n}: {d}" for n, d in nodes.items()] if nodes else ["(none yet)"]
    L += ["", "New records:"]
    for i, rec in enumerate(chunk):
        L.append(f"[record {start + i}]")
        L += [f"{k}={v}" for k, v in rec.items()]
        L.append("")
    L += ["For every record above, decide which node it attaches to. Output a JSON
    array "
          "with one object per record, in record order:",
          '- existing node: {"record": <id>, "node": "<existing node name, exactly
    as listed>"}',
          '- new node: {"record": <id>, "node": "<new node name>", "new": true, '
          '"description": "<one line saying what kind of item the node stands for>"}
    ',
          '- background: {"record": <id>, "action": "background"}',
          "A node created for an earlier record in this same answer may be reused by
     later "
          "records by name. Output the JSON array only."]
    return "\n".join(L)
```

H.4 Matched Wikidata request and parsing contract

Each model receives 30 requests per stream, with 20 prefix records per request. There are two feature variants, five sampled orders, a signal and a shuffled control, and zero-shot and few-shot conditions: 40 streams and 1,200 requests per model. Only a transport failure permits a retry. A model response that reaches the output limit or fails the response contract is retained as returned. The following settings and parser specify how these responses become groups. Exact requests are stored under `evidence/matched_prediction/llm/`, indexed by model, prompt condition, input identifier, and chunk number.

```
{
  "requests_per_stream": 30,
```

```
  "records_per_request": 20,
  "streams_per_model": 40,
  "prompt_builder": "real_prompt_builder.py",
  "gemini": {
    "model": "gemini-3.8-flash",
    "temperature": 0,
    "reasoning_effort": "low",
    "max_tokens": 8192,
    "system_message": null
  },
  "qwen": {
    "model": "Qwen/Qwen2.5-14B-Instruct",
    "do_sample": false,
    "precision": "bfloat16",
    "max_new_tokens": 8192,
    "batch_size": 4,
    "padding_side": "left",
    "system_message": "Tokenizer chat template default",
    "checkpoint_revision": "cf98f3b3bbb457ad9e2bb7baf9a0125b6b88caa8",
    "torch": "2.5.1+cu121",
    "transformers": "5.16.1",
    "gpu": "NVIDIA A100-SXM4-80GB",
    "eos_token_id": [
      151645,
      151643
    ],
    "pad_token_id": 151643
  },
  "parser": "matched_wikidata_parser.py",
  "retry_policy": "Up to two transport retries per attempt; interrupted streams
    resume from exact cached requests. No model-output repair or quality retry."
}
```

```
import json, re, collections

def parse(raw, start, size, nodes, members):
    text=raw.strip()
    if text.startswith('```'):
        text=re.sub(r'^```(?:json)?\s*','',text);text=re.sub(r'\s*```$','',text)
    report=dict(expected=size,valid=0,background=0,missing=0,duplicates=0,
    out_of_chunk=0,invalid=0,parse_error=False)
    try:
        values=json.loads(text)
        if not isinstance(values,list): raise ValueError('not an array')
    except (ValueError,TypeError):
        report.update(parse_error=True,missing=size);return report
    byid=collections.defaultdict(list)
    for row in values:
        if not isinstance(row,dict) or type(row.get('record')) is not int:
            report['invalid']+=1;continue
        rid=row['record']
        if not start<=rid<start+size:report['out_of_chunk']+=1;continue
        byid[rid].append(row)
    for rid in range(start,start+size):
        rows=byid.get(rid,[])
        if not rows:report['missing']+=1;continue
        if len(rows)!=1:report['duplicates']+=1;continue
        o=rows[0]
        if o.get('action')=='background' and not o.get('node'):
            report['valid']+=1;report['background']+=1;continue
        name=o.get('node')
        if not isinstance(name,str) or not name.strip() or o.get('action'):
            report['invalid']+=1;continue
        name=name.strip()
```

```
        if o.get('new') is True and name not in nodes:
            if not isinstance(o.get('description'),str) or not o['description'].
    strip():
                report['invalid']+=1;continue
            nodes[name]=o['description'][:120]
        if name not in nodes:report['invalid']+=1;continue
        members.setdefault(name,set()).add(rid);report['valid']+=1
    return report
```

H.5 ARCHIVED MATCHED HIERARCHY INSTRUCTION

This protocol supplies one new record per arrival, permits abstention and deferral, and allows graph repair at declared boundaries. It does not force nonempty output.

```
Discover recurring groups and how they nest from categorical records.
Key and value names are opaque tokens. You have no supplied taxonomy, category
count, depth or mapping of fields to levels. Infer only what the observations
support. A parent may have children and itself be a child. Multiple parents are
allowed for joint specialisation; ordinary overlap does not imply ancestry.
Preserve useful parents. Leave uncertain links absent and describe their candidate
node IDs in unresolved. Child members must also be members of every parent.
Do not create cycles or transitively redundant direct parent links.

Each arrival supplies exactly one new record. Retain previous observations in
this conversation. Graph changes are allowed only when repair_allowed is true:
every 100 arrivals and one terminal repair with no new records. At other arrivals
return nodes:null and action background or defer. You may maintain brief notes.
At a repair you may create, delete, merge, split, or revise memberships and links
for ANY observed record, and may also leave everything in background. No unseen
record may be referenced. Keep existing node IDs for surviving groups; never
reuse a deleted ID. There is no implicit requirement to create any groups.

Return a JSON object with action (background, defer or revise), nodes (null for
unchanged, or the complete replacement graph), unresolved (list of brief strings),
and notes (brief observations). A node is an object with positive integer id,
members (observed zero-based record IDs), support (observed key strings), and
parents (direct parent IDs). Explicitly include inherited ancestor memberships.
Background means currently ungrouped; defer means an unresolved decision. An
invalid answer is recorded separately and does not become an abstention.
```

GENERATION SETTINGS.

```
{
  "model": "gemini-3.8-flash",
  "temperature": 0.0,
  "reasoning_effort": "low",
  "max_tokens": 16384,
  "response_format_type": "json_schema",
  "schema_name": "hierarchy_update",
  "strict": true,
  "tools_supplied": [],
  "system_utf8_sha256": "
    c1d6da057f6833c6b2d69854a6d9bcb1d4fd5a2b9d880edda4d857a3a6583477",
  "E81_source_sha256": "
    ac883d65b22217f2656076e9ab8254c11f5c429025e16dc5197ddec2e150082d",
  "request_message_role": "system once, then cumulative user/assistant conversation
    ",
  "new_records_per_arrival": 1,
  "terminal_new_records": 0
}
```

AN ACTUAL FIRST-ARRIVAL MESSAGE.

```
{
  "prefix": 1,
  "new_record": {
    "id": 0,
    "fields": {
      "ff8a92adef411": "v9e28bbe4080d",
      "fab37996b0150": "vb7c68997c164",
      "f1ce9dd3a9f34": "v19b50c649a4d",
      "f377515d3cf47": "v1fb90d408c06",
      "f1792bb3a8197": "v2fa64b859885",
      "f9eea298e5865": "v24f4d38dd297",
      "f2f9cdc0cea75": "vd1d07bfae8e9",
      "f8eafacae349f": "v02ea6543151d",
      "fba210eacbf7c": "v67b5780fc81b",
      "f235aef57142e": "vcc0b4a5ae8dd",
      "f6b27a6b334d7": "ve3e28a33fd5f",
      "fbb520d2f8ab5": "ve9e76f1723af"
    }
  },
  "repair_allowed": false,
  "terminal_repair": false,
  "graph": [],
  "graph_unchanged_since_last_repair": true,
  "next_unused_node_id": 1
}
```

AN ACTUAL TERMINAL MESSAGE FROM A NULL STREAM.

```
{
  "prefix": 300,
  "new_record": null,
  "repair_allowed": true,
  "terminal_repair": true,
  "graph": [],
  "graph_unchanged_since_last_repair": false,
  "next_unused_node_id": 1
}
```

COMPLETE RESPONSE SCHEMA.

```
{
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "action": {
      "type": "string",
      "enum": [
        "background",
        "defer",
        "revise"
      ]
    },
    "nodes": {
      "anyOf": [
        {
          "type": "null"
        },
        {
          "type": "array",
          "items": {
            "type": "object",
```

```
            "additionalProperties": false,
            "properties": {
              "id": {
                "type": "integer"
              },
              "members": {
                "type": "array",
                "items": {
                  "type": "integer"
                }
              },
              "support": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              },
              "parents": {
                "type": "array",
                "items": {
                  "type": "integer"
                }
              }
            },
            "required": [
              "id",
              "members",
              "support",
              "parents"
            ]
          }
        }
      ]
    },
    "unresolved": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "notes": {
      "type": "string"
    }
  },
  "required": [
    "action",
    "nodes",
    "unresolved",
    "notes"
  ]
}
```

H.6 Compact-context depth instruction and request construction

The compact context version retains the current graph and reveals the observed prefix at permitted repair steps. The source below fixes both the system text and the changing user payload. The matched campaign's output allowance is set by its own protocol.

```
Discover recurring groups and how they nest from categorical records.
Key and value names are opaque tokens. You have no supplied taxonomy, category
count, depth or mapping of fields to levels. Infer only what the observations
support. A parent may have children and itself be a child. Multiple parents are
allowed for joint specialisation; ordinary overlap does not imply ancestry.
Preserve useful parents. Leave uncertain links absent and describe their candidate
```

```
node IDs in unresolved. Child members must also be members of every parent.
Do not create cycles or transitively redundant direct parent links.

Each arrival supplies exactly one new record. Use the retained observations supplied
     at repair. Graph changes are allowed only when repair_allowed is true:
every 100 arrivals and one terminal repair with no new records. At other arrivals
return nodes:null and action background or defer. You may maintain brief notes.
At a repair you may create, delete, merge, split, or revise memberships and links
for ANY observed record, and may also leave everything in background. No unseen
record may be referenced. Keep existing node IDs for surviving groups; never
reuse a deleted ID. There is no implicit requirement to create any groups.

Return a JSON object with action (background, defer or revise), nodes (null for
unchanged, or the complete replacement graph), unresolved (list of brief strings),
and notes (brief observations). A node is an object with positive integer id,
members (observed zero-based record IDs), support (observed key strings), and
parents (direct parent IDs). Explicitly include inherited ancestor memberships.
Background means currently ungrouped; defer means an unresolved decision. An
invalid answer is recorded separately and does not become an abstention.

Context protocol: each request contains one new record, or no new record at terminal
     repair. Only your latest notes persist between requests as scratch memory.
    Keep notes concise. The caller retains all observations. At each allowed repair,
     prior_observations supplies every earlier record, with the current new record
    separate. Use the supplied graph as the current accepted state. There are no
    future records, examples, category names, group counts or target levels. At non-
    repair arrivals the accepted graph is unchanged.
```

```
def request(state,record,terminal,notes):
 m=make_message(state,record,terminal);m['previous_notes']=notes
 if m['repair_allowed']:
  end=len(state.records) if terminal else len(state.records)-1
  m['prior_observations']=[{'id':i,'fields':state.records[i]} for i in range(end)]
 return [{'role':'system','content':COMPACT_SYSTEM},{'role':'user','content':json.
    dumps(m,separators=(',',':'))}]
```

H.7 Prompt-sensitivity variants

The following complete prompt variants are retained for transparency about prompt development. Their outcomes are not pooled into the matched real-data or matched-depth comparisons. The shipped instruction and the alternative wordings are distinct interventions; choosing the most favorable one after observing a result would be another selection step.

Single-record system instruction

```
You build a graph of nodes from a stream of records. Each record describes one item
    as key=value lines. A node stands for one kind of item. Records of the same
    kind of item attach to the same node. Reuse an existing node whenever one fits.
     Create a new node only when no existing node fits. Reply with JSON only.
```

Default instruction (e42)

```
You build a graph of nodes from a stream of records. Each record describes one
item as key=value lines. A node stands for one kind of item. Records of the same
 kind of item attach to the same node. Reuse an existing node whenever one fits.
 Create a new node only when no existing node fits. Reply with JSON only.
```

Persona paraphrase (e42)

```
You are an experienced data engineer who organises catalogues. You build a graph
 of nodes from a stream of records. Each record describes one item as key=value
lines. A node stands for one kind of item. Records of the same kind of item
attach to the same node. Reuse an existing node whenever one fits. Create a new
node only when no existing node fits. Reply with JSON only.
```

REORDERED INSTRUCTION (E42)

```
Reply with JSON only. You are building a graph of nodes from a stream of
records, where each record describes one item as key=value lines. One node
stands for one kind of item, and records describing the same kind of item attach
 to the same node. Whenever an existing node fits, reuse it; create a new node
only when none fits.
```

BULLETED INSTRUCTION (E42)

```
You build a graph of nodes from a stream of records.
- Each record describes one
 item as key=value lines.
- A node stands for one kind of item.
- Records of the
 same kind of item attach to the same node.
- Reuse an existing node whenever
one fits.
- Create a new node only when no existing node fits.
- Reply with JSON
 only.
```

LIBERAL GROUP CREATION (E42)

```
You build a graph of nodes from a stream of records. Each record describes one
item as key=value lines. A node stands for one kind of item. Records of the same
 kind of item attach to the same node. Be precise: create a new node whenever a
record does not clearly match an existing one, so that each node stays a single
well defined kind. Reply with JSON only.
```

CONSERVATIVE GROUP CREATION (E42)

```
You build a graph of nodes from a stream of records. Each record describes one
item as key=value lines. A node stands for one kind of item. Records of the same
 kind of item attach to the same node. Be conservative: strongly prefer reusing
an existing node, and keep the number of nodes small. Create a new node only
when no existing node could possibly fit. Reply with JSON only.
```

ALLOWING NO STRUCTURE (E42)

```
You build a graph of nodes from a stream of records. Each record describes one
item as key=value lines. A node stands for one kind of item. Records of the same
 kind of item attach to the same node. Reuse an existing node whenever one fits.
 Some streams have no kinds in them at all, in which case every record belongs
to one single node and you should create no others. Create a new node only when
no existing node fits. Reply with JSON only.
```

I REPRODUCTION, PROVENANCE, AND VERIFICATION

Evidence files. The source package contains `evidence/` JSON files for every figure and table, a source manifest with SHA-256 hashes, and selected protocol records. Each result family remains separate. Source inspection is recorded with its own snapshot and date. A hash of source inspected for the source snapshot is not claimed to be the historical executable hash of a run that did not record one.

Rebuilding the figures and tables. Install the dependencies in `requirements.txt` and run `bash rebuild.sh` from the extracted supplementary directory. The script composes the standard-baseline results, then regenerates figures and tables from saved outcomes without model calls or experiment reruns. The matched figures read `evidence/standard_baselines/derived/matched_prediction/report/`. The archive supplies code and evidence; the compiled manuscript is submitted separately. Figure axes, units, reference conditions, and sample counts are specified in the text and captions.

Verification. Nine mathematical and implementation checks cover conditional coding and fast model costs. They check empty-graph overhead, record-order invariance, enumerated Kraft sums, type-class arithmetic, the evidence margin, scalar versus array scoring, search charges, and the escape code. All passed.

The supporting audits preserve each experiment's protocol and inputs. The pruning study supplies 126 outcomes and 84 reconstructed certificates. The parent-based study supplies 175 cases and independent direct-probability checks. The same-input implementation audit adds 42 six-arm cases, two cold replays, and 168 reconstructed certificates. COBWEB supplies 42 cases, 84 terminal outcomes, and 168 reconstructed checkpoint decisions. The matched conditional comparison contains 504 native, whole, and pruned outcomes, with 336 certificates checked by scalar and vectorized code. Recovery matching, equally optimal matches, graph indices, ownership lists, block hashes, and score calculations are retained. Appendix E.7.1 reports readout sensitivity; Appendix E.8 reports the conditional predictor's float32 differences. These checks establish the stated measurement properties, not the correctness of every possible input or execution.

Matched prediction evidence. The joint comparison supplies conventional prediction and conditional compression for native and selected graphs on 82 fixed inputs. Wikidata includes both ESCROW searches, COBWEB, FISHDBC, eight other classical graph proposers, four LLM conditions, and prediction-only KNN. Every LLM request, raw response, parser diagnostic, and model identifier is retained. Five prediction-head settings separate semantic sharing from adaptive weighting. Threshold replay covers 4,086 decisions. Source hashes, scalar prediction checks, normalization tests, and paired outcome tables link the reported means to their inputs.

DP-means has 238 independent small-input comparisons and four full-development comparisons against a dense implementation. BIRCH checks include actual tree splits. The complete parameter sweep contains 4,658 fits on development and noise inputs; all DP-means runs converge with centroids equal to assigned-record means. On Wikidata, 40 scalar certificate reconstructions and 240 prediction-block checks cover both methods and both selection conditions. Their complete grids, selected settings, inputs, source hashes, and verification records are packaged under `evidence/standard_baselines/`.

Protocol boundaries. The matched Wikidata study has complete request logs. The separate single-block LLM study retains its builder and output summaries only; its coverage limitations are stated with that table. Fixed models predict three future blocks in the matched study. Continuously changing graphs and memory weights carried across blocks are separate research questions. The four feature variants come from two corpora.

Interpretation boundaries. Each guarantee applies to its declared code and null. Native recursive selection and the conditional gate are distinct. Whole-graph evidence does not certify every recovered group or parent link.