THE STRUCTURAL GAIN IN TEMPORAL LINK PREDICTION IS MOSTLY MEMORY AND RECENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal graph networks increasingly encode neighborhood structure, such as neighbors that co-occur in two nodes' recent histories, shared neighbors inside a sliding window, or temporal walks. The premise is that structure reveals who will interact next. We show that a structural feature computed from an event history also carries two other kinds of information: memory, whether the pair has ever interacted, and recency, when its endpoints were last active. The memory part is provable. Distance is the first depth at which two neighborhoods overlap, and the overlap vector is supported on the three depths around it, so a non-zero overlap at three hops certifies that a pair has never met. We measure the three kinds of information on 14 benchmarks under the random and historical negative protocols of DyGLib. Our test admits a structural feature only after memory, and then only after recency, and checks the result against a shuffled control and a second scorer. Under historical negatives, neighborhood overlap, the classical heuristics and DyGFormer's co-occurrence encoding keep a gain above 0.01 AP after memory on 0, 1 and 9 of the 14 benchmarks, and after recency on 0, 0 and 0. Across four structural signals, a gain remains in 14 of 112 benchmark-protocol cells, with median +0.017. Inside a trained model the picture is the same. Nine numbers about the pair's own history, with no neighbor of either node, replace DyGFormer's co-occurrence channel on 10 of 13 benchmarks, and the two exceptions are the two benchmarks on which the test found structure surviving. Structure survives where memory cannot answer: where no pair repeats (on a repeat-free benchmark overlap adds +0.121 and keeps +0.120 after recency), where every pair repeats (a dense trade network), and in the cold part of ordinary benchmarks. Gating a structural feature to the regime the test identifies keeps a median of 98% of its gain on 66% of queries. We release the test as `structaudit`, together with measurements of three construction mistakes that silently change its answer.

1 INTRODUCTION

Temporal link prediction asks which pair of nodes will interact next in a stream of timestamped events. Many recent models answer with the structure around the candidate pair. DyGFormer counts how often each neighbor appears in the recent histories of both endpoints (Yu et al., 2023). CAWN samples temporal walks between them (Wang et al., 2021). TNCN and TPNet carry common neighbors and walk matrices into the temporal setting (Zhang et al., 2024; Lu et al., 2024). Hawkeye maintains the k-core cohesion of the two neighborhoods inside a sliding window (Ding & Zhang, 2026). Each of these models is compared with the same backbone without its structural channel, and the gain is read as evidence that structure predicts who will interact next.

A structural feature computed from an event history carries more than structure. It reveals whether the two nodes have ever interacted, because a pair that has met is at distance one, and a pair at distance one has no overlap at three hops (Corollary 1). A feature that can express "never met" is doing the work of memory. The same feature also moves with how recently the endpoints were active, and a sliding window ties it to the recent past by construction. The comparison with and without the structural channel therefore cannot separate structure from memory and time.

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.

We separate the three by what a feature reads. Memory reads the events of the pair and of the candidate, and whether they exist at all. Recency reads when those events happened. Structure reads the neighbors of the endpoints. We replay 14 temporal link-prediction benchmarks in strict time order under the random and historical negative protocols of DyGLib (Yu et al., 2023), with three seeds each, and we admit a structural feature only after memory, and then only after recency. The apparent structural gain disappears in that order. Neighborhood overlap adds up to $+0.113$ AP to popularity and the pair count under historical negatives. After memory, a gain that two scorers agree on remains on 0 of 14 benchmarks. The classical heuristics keep a gain on 1 benchmark after memory and on 0 after recency. DyGFormer's co-occurrence encoding, the part its authors single out as structural, keeps a gain on 9 benchmarks after memory and on 0 after recency.

The same result holds inside a trained model. Without its co-occurrence channel, DyGFormer loses a median of +0.060 AP under one training budget per benchmark. Nine numbers about the pair's own history, which read no neighbor of either node, give that loss back on 10 of 13 benchmarks (Section 4).

Structure does not vanish everywhere, and the places where it survives share a property. Across the four signals on every benchmark, a gain that survives memory, recency, a shuffled-feature control and a second scorer remains in 14 of 112 cells, with median +0.017 AP. The larger residuals sit at the two ends where memory cannot answer. On UN Trade every pair repeats, so memory marks almost every candidate as a partner and recency adds only +0.014, while shared-partner counts still add +0.038. On the TGB-Seq GoogleLocal stream (Yi et al., 2025) no pair repeats, and overlap adds +0.121 and keeps +0.120 after recency. The standard benchmarks sit between these two ends, and there the structural gain is mostly memory and time.

We package the analysis as a test to be run before a structural claim is made. The test takes an edge stream and any structural feature, replays the stream causally under DyGLib's protocols, and reports what the feature adds after memory, after recency and against a shuffled control, on a CPU. Building it showed us three construction choices that silently change the answer, and we measure each of them.

Contributions. We separate memory, recency and structure by what a feature reads, and we prove two propositions that explain why a structural feature carries memory (Section 5). We measure the three on 14 benchmarks, two negative protocols and four structural signals, including the encodings of two published models, with a shuffled control and a second scorer (Section 3). We confirm the result inside two trained models (Section 4). We show where structure survives, on the repeat-free benchmark, on the saturated one and in dense sessional networks, and we show a protocol that cannot see any of it (Section 6). We release the test as `structaudit`, and we measure the three construction mistakes it prevents (Section 2).

2 Separating Memory, Recency and Structure

Protocol. A benchmark is a stream of events (u, v, t). We follow DyGLib, the library and evaluation protocol released with DyGFormer (Yu et al., 2023). The stream is split at the 0.70 and 0.85 quantiles of time into training, validation and test periods, and every validation and test event is paired with one negative. Under the random protocol the negative replaces the destination by one drawn uniformly from the destinations of the whole dataset. Under the historical protocol the negative is an edge seen before the current batch and absent from it; such a negative has occurred before, so repetition alone cannot separate it from a repeated positive. Queries are processed in batches of 64 events, and a batch sees only the events strictly earlier than its first event. A DyGLib model sees every event strictly before the query's own time, so our history is never larger than the model's, and events that share a timestamp never see one another.

Three kinds of information. We sort features by what they read, not by what they are called. Take a query (u, v, t) with candidate v, with every quantity computed from events strictly before t on the undirected history graph. *Memory* (M) reads the events of the pair and of the candidate, and whether they exist. Its columns are $\log(1+\deg v)$, $\log(1+$number of events of $v)$ and $\log(1+$number of past events between u and $v)$, and two indicators: that the pair has never met, and that the candidate has never been active. *Recency* (T) reads when those events happened: $\log(1 + t - t_{uv})$ with t_{uv} the

last time u and v interacted, and $\log(1 + t - t_v)$ with t_v the last time v was active. *Structure* reads the neighbors of the endpoints, in three blocks. The distance block (D) holds indicators of the first depth $h \leq 3$ with $\mathrm{ov}_h(u, v) > 0$. The overlap block (O) holds $\log(1 + \mathrm{ov}_h(u, v))$ for $h = 1, 2, 3$, with $\mathrm{ov}_h(u, v) = |\mathcal{N}(v) \cap \mathcal{N}_h(u)|$ and $\mathcal{N}_h(u)$ the nodes at distance exactly h from u. The feature under test (S) is any function of the history and the pair. The two indicators belong to memory rather than to recency, because they say whether an event ever happened and not when. Kept with recency, they would credit time with work that memory does, and Corollary 1 shows that a structural feature can do that work too.

The test. Every block combination is scored by a standardized logistic regression, fitted on the validation period and evaluated by average precision (AP) on the test period. For a structural feature S the test reports what it adds at each step:

$$\begin{aligned}
\Delta_{\text{alone}} &= \mathrm{AP}(\mathrm{P{+}R{+}S}) - \mathrm{AP}(\mathrm{P{+}R}), \quad \text{popularity and the pair count only,} \\
\Delta_{\text{memory}} &= \mathrm{AP}(\mathrm{M{+}S}) - \mathrm{AP}(\mathrm{M}), \quad \text{memory, including whether the pair ever met,} \\
\Delta_{\text{recency}} &= \mathrm{AP}(\mathrm{M{+}T{+}S}) - \mathrm{AP}(\mathrm{M{+}T}).
\end{aligned}$$

We read a gain as memory when $\Delta_{\text{alone}} > 0.01$ and $\Delta_{\text{memory}} \leq 0.01$, as recency when $\Delta_{\text{memory}} > 0.01$ and $\Delta_{\text{recency}} \leq 0.01$, and as structure when $\Delta_{\text{recency}} > 0.01$. The same differences are reported within each evidence regime: the pair has met before, both endpoints are known but the pair is new, or an endpoint is new.

Two controls. A feature with many columns can raise AP by giving the scorer more freedom, and a linear scorer can miss what a feature expresses nonlinearly. The first control permutes the rows of the feature block in the validation and test matrices independently and repeats the measurement, so whatever a shuffled feature appears to add is the noise floor for its dimension on that benchmark. Across the 14 benchmarks, both protocols and every structural feature, a shuffled block adds a median of +0.0000 AP. The largest floor is +0.018, on US Legis., whose validation and test periods fall in different parliaments. Every residual reported below exceeds the floor measured on its own benchmark. The second control rescores every matrix with gradient-boosted trees; the two scorers agree on the verdict in 98 of the 112 cells. We count a structural gain only where both scorers see it. The 14 cells they disagree on include the largest single residual in the study, Can. Parl. under random negatives, which the linear scorer puts at +0.103 and the trees at -0.058.

Plugging in a feature. A structural feature is an object with two methods: one receives events once time has moved past them, the other returns a vector per query pair. The test owns the replay, so a feature cannot see the query pair or anything simultaneous with it.

Three construction mistakes. Each of the following changed our own measurements before we caught it, and Table 1 reproduces each with the test's own code, one at a time. The first is simultaneous events. On benchmarks recorded by year or day, most events share a timestamp with others (all of them on six of the 14). Replaying them in file order lets an event see the others of its timestamp, which a DyGLib model cannot. The floor on UN Vote then rises from 0.539 to 0.766, and overlap on US Legis., worth +0.000 on its own, appears to be worth +0.158. The second is the negative pool. Random negatives drawn uniformly over node identifiers, rather than over destinations, are mostly impossible destinations on bipartite streams, and they lift the MOOC floor from 0.760 to 0.995. The third is the fitting period. A scorer fitted on the first half of the test period instead of the validation period, and evaluated on the second half like the others, lifts the Can. Parl. floor from 0.693 to 0.816. The regime of a parliament changes between sessions, and a scorer fitted after the change is fitted to the answer.

3 What Structure Adds Once Memory and Recency Are In

We test four structural signals, each reproduced from its authors' definition and code. *Neighborhood overlap* is the block O above. The *classical heuristics* are common neighbors, Adamic-Adar, resource allocation and Jaccard on the cumulative history graph (Liben-Nowell & Kleinberg, 2007). *DyGFormer's neighbor co-occurrence* (Yu et al., 2023) takes each endpoint's sequence, the node followed by its most recent $L - 1$ neighbors, with L the value DyGLib tunes per benchmark (from

Table 1: Each construction mistake, switched on alone. Random negatives, mean of three seeds. The floor is the logistic regression on memory, recency, distance and overlap; "overlap alone" is its gain over popularity and repetition.

		floor AP		overlap alone	
mistake	benchmark	correct	mistaken	correct	mistaken
simultaneous events visible	UN Vote	0.539	**0.766**	−0.001	−0.003
	Can. Parl.	0.719	**0.918**	−0.009	+0.009
	US Legis.	0.654	**0.825**	+0.000	+0.158
	UN Trade	0.682	**0.832**	+0.041	+0.044
negatives over all node ids	MOOC	0.760	**0.995**	−0.000	+0.008
	Wikipedia	0.979	**0.996**	−0.000	+0.002
	Reddit	0.991	**0.999**	+0.005	+0.002
scorer fitted inside the test period	Can. Parl.	0.693	**0.816**	−0.004	+0.009
	US Legis.	0.655	**0.707**	+0.003	+0.006

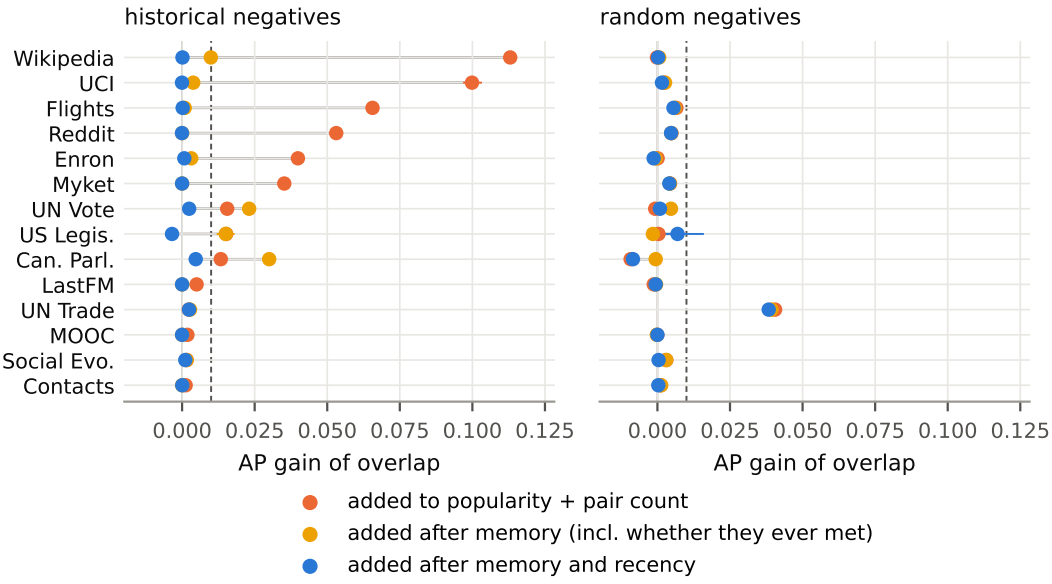

Figure 1: What neighborhood overlap adds, per benchmark, as the model is given more of what is not structure: popularity and the pair count only, then memory (which includes whether the pair ever met), then recency. Means of three seeds, whiskers the seed range; the dashed line marks 0.01.

32 to 2048). Every position is annotated with how often its node occurs in the two sequences. The model reads these counts with an MLP and a transformer; the test reads, per side, their total and the number of positions with a non-zero cross count. *Hawkeye's cohesion channel* (Ding & Zhang, 2026) is the 23 pairwise features of its released cache in its published configuration (degree and k-core indicators, full backend, a window of 1% of the time span), for the pair in both directions.

Overlap after memory. Figure 1 follows overlap as memory and then recency enter the model. Added to popularity and the pair count, overlap is worth up to +0.113 AP under historical negatives. After memory, it keeps a gain that both scorers see on 0 of 14 benchmarks. The size of this step follows from Corollary 1. The overlap vector reveals whether the pair has ever met, and under historical negatives every negative is a pair that has met, so that one bit is most of the answer. The classical heuristics behave the same way: they keep a gain on 1 of 14 benchmarks after memory and on 0 after recency.

Table 2: How many of the 14 benchmarks each structural signal helps on, after memory and after recency, and the largest gain in each case. A benchmark counts when the mean gain over three seeds exceeds 0.01 AP under both scorers and, after recency, also exceeds the shuffled floor of that benchmark.

		helps on ($>$ 0.01 AP)		largest gain	
negatives	structural feature	after mem.	after rec.	after mem.	after rec.
random	overlap	1/14	**1/14**	+0.039 (UN Trade)	+0.038 (UN Trade)
	classical heuristics	5/14	**2/14**	+0.043 (UN Trade)	+0.039 (UN Trade)
	DyGFormer co-occ.	7/14	**3/14**	+0.069 (MOOC)	+0.028 (UN Trade)
	Hawkeye cohesion	9/14	**4/14**	+0.137 (MOOC)	+0.051 (MOOC)
historical	overlap	0/14	**0/14**	–	–
	classical heuristics	1/14	**0/14**	+0.098 (US Legis.)	–
	DyGFormer co-occ.	9/14	**0/14**	+0.163 (Contacts)	–
	Hawkeye cohesion	11/14	**4/14**	+0.157 (Can. Parl.)	+0.040 (Can. Parl.)

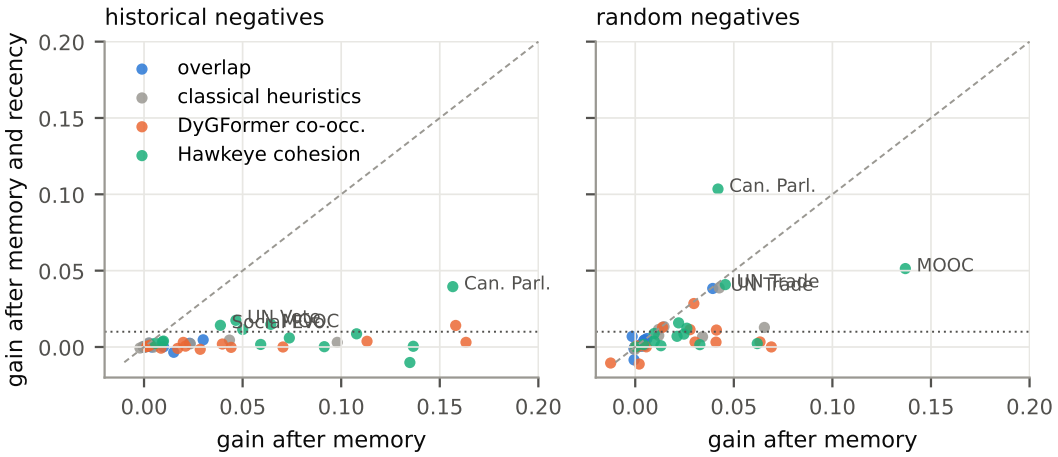

Figure 2: Every benchmark and structural signal: what it adds after memory (horizontal) against what it still adds after recency (vertical). Points on the diagonal are untouched by recency, points on the floor are recency in disguise. Labels mark the cells above 0.01 after recency.

Recency. What memory leaves is largely recency. The two recency times add a median of +0.106 AP over memory under historical negatives and +0.025 under random ones (Table 2). The historical protocol explains the larger number. A historical negative is drawn from all edges seen so far, so it is typically an old edge, while the positive is happening now. Once memory has established that both pairs have met, the time since they last met is what separates them, and any feature that varies with the endpoints' recent activity inherits that separation. DyGFormer's co-occurrence encoding is the part of the model its authors single out as structural. Under this protocol it keeps a gain on 9 of 14 benchmarks after memory and on 0 after recency; under random negatives it keeps a gain on 3 (Table 2).

The residual. Hawkeye's cohesion channel is the exception among the four. It keeps a gain on 11 benchmarks after memory and on 4 after recency under historical negatives, and on 4 under random negatives (Table 2). Its features are 46-dimensional against three for overlap, but capacity is not the reason: every one of these residuals exceeds the shuffled floor of its benchmark. Count all four signals on all 14 benchmarks. A gain that survives memory, recency, the shuffled control and both scorers remains in 4 of 56 cells under historical negatives and 10 under random ones, 14 of 112 in total (Figure 2). The median surviving gain is +0.017 AP, and the two largest are MOOC (hawkeye, +0.051) and UN Trade (hawkeye, +0.041). Four fifths of what looks structural on these benchmarks is memory or recency. What remains is small and concentrated, and Section 6 asks where it lives; the per-benchmark numbers are in the supplementary material.

Table 3: Test AP of four DyGFormer variants under one budget per benchmark (epochs and number of seeds in the second column; mean and standard deviation where there is more than one seed): the published neighbor co-occurrence channel, no structural channel, the pair-time channel alone, and both. The pair-time channel reads only the events of the pair and its endpoints. Standard deviations are given in units of the last digit, so $0.9903(2)$ means 0.9903 ± 0.0002. The best of the four is in bold; the last column is the pair-time channel alone minus the published model, in bold where it is within 0.005 or above. UN Trade did not complete.

benchmark	ep., seeds	co-occ.	none	pair-time	both	pair-time − co-occ.
Wikipedia	20, 3	$0.9903(2)$	$0.9418(17)$	$0.9900(2)$	$\mathbf{0.9923(3)}$	$\mathbf{0.000}$
Reddit	20, 2	$0.9921(1)$	$0.9690(0)$	$0.9911(0)$	$\mathbf{0.9929(1)}$	$\mathbf{-0.001}$
MOOC	20, 3	$0.8546(36)$	$0.7943(88)$	$0.8574(14)$	$\mathbf{0.8652(21)}$	$\mathbf{+0.003}$
LastFM	4, 2	$0.9193(2)$	$0.5351(63)$	$0.9211(1)$	$\mathbf{0.9284(4)}$	$\mathbf{+0.002}$
Myket	20, 2	$0.8022(1)$	$0.6955(86)$	$0.8812(0)$	$\mathbf{0.8843(3)}$	$\mathbf{+0.079}$
Enron	20, 3	$0.9219(12)$	$0.6548(164)$	$0.9249(17)$	$\mathbf{0.9263(10)}$	$\mathbf{+0.003}$
Social Evo.	8, 2	$0.9458(0)$	$0.9311(1)$	$0.9439(6)$	$\mathbf{0.9469(0)}$	$\mathbf{-0.002}$
UCI	20, 3	$0.9568(9)$	$0.8409(53)$	$0.9627(5)$	$\mathbf{0.9676(12)}$	$\mathbf{+0.006}$
Flights	4, 2	$0.9887(1)$	$0.9097(20)$	$0.9758(2)$	$\mathbf{0.9890(1)}$	-0.013
Can. Parl.	20, 1	$\mathbf{0.9784}$	0.9574	0.9367	0.9713	-0.042
US Legis.	20, 2	$0.7031(107)$	$\mathbf{0.7060(37)}$	$0.7039(122)$	$0.6977(11)$	$\mathbf{+0.001}$
UN Vote	8, 1	0.5551	0.5235	0.5415	$\mathbf{0.5649}$	-0.014
Contacts	8, 2	$0.9821(3)$	$0.9198(11)$	$0.9780(1)$	$\mathbf{0.9834(0)}$	$\mathbf{-0.004}$

4 Inside a Trained Model

The test reads features through simple scorers, and a trained model might use a structural channel in ways a summary does not express. We therefore repeat the question inside DyGFormer. The *pair-time channel* is nine numbers for a candidate pair. Six are the log time since the pair last interacted, since the candidate was last active and since the source was last active, each with an indicator that it never happened; three are the log counts of those events. The channel reads events strictly before the query, the rule of DyGLib's neighbor sampler, and no neighbor of either node; a two-layer MLP adds its output to the logit of DyGLib's link predictor. Four variants are trained with everything else equal: DyGLib's tuned configuration for the benchmark, one epoch budget per benchmark, patience five, the same seeds and random negatives. The budget is twenty epochs, reduced to eight or four on the largest streams so that all four variants could finish. The variants are the published co-occurrence channel, no structural channel, the pair-time channel alone, and both (Table 3). Our co-occurrence baseline reproduces the published model to within 0.003 AP on UCI, Enron, Wikipedia, Reddit and Flights (0.958, 0.925, 0.990, 0.992 and 0.989 in Yu et al. (2023)); on MOOC and LastFM the reduced budget leaves it 0.02 and 0.01 below.

Across the 13 benchmarks, the co-occurrence channel is worth a median of +0.060 AP over no channel, and this is the comparison that credits it with structure. The pair-time channel gives that gain back. On 10 of the 13 benchmarks the pair-time channel alone comes within 0.005 of the published model or exceeds it (last column of Table 3), and the co-occurrence channel on top of it is worth a median of +0.005. The two exceptions are Can. Parl. and Flights, the two streams on which the feature-level test had found structure surviving memory and recency (Section 3). Inside the trained model, as in the test, the structural channel contributes almost entirely memory and recency.

Another backbone. The pair-time channel is not tied to DyGFormer. We add it to TGN, a memory-based model with no structural channel, on UCI and Enron, with DyGLib's tuned configurations and the same budget (per-seed numbers in the supplementary material). On UCI the channel raises TGN from 0.888 to 0.964 AP, and on Enron from 0.843 to 0.921, gains of +0.076 and +0.078. A backbone that reads no neighborhood structure reaches the level of the published DyGFormer once it is told when the pair and its endpoints were last active.

5 WHERE THE TIME ENTERS

Windows. Hawkeye's 23 features split into ten that describe the current graph (cohesion-weighted common neighbors and two-hop bridges, and the core numbers of the endpoints) and thirteen that describe how those core numbers have been changing. Crossing this split with the window shows where the recency in them comes from. Take UCI under historical negatives, after memory. The ten structural features are worth +0.014 AP when computed on the whole history and +0.086 when computed inside Hawkeye's window of 1% of the time span; on Wikipedia the two numbers are +0.009 and +0.054. The same structure becomes several times more predictive once it is restricted to the recent past, and recency then removes the gain. The dynamics features are a second route to the same information: on the cumulative graph all 23 features are worth +0.047 on UCI, against +0.014 for the structural ten. Hawkeye's authors note that a k-core decomposition is itself an adaptive temporal filter, because a node that stops interacting is eventually peeled to a lower core (Ding & Zhang, 2026). The test measures how much of the filter's output is the time it filters by.

Distance. Distance, the most compact structural summary, is tied to overlap by an identity.

Proposition 1. *For any graph and any $v \neq u$, with $\mathcal{N}_0(u) = \{u\}$, $d(u,v) = 1 + \min\{\, h \geq 0 : \mathrm{ov}_h(u,v) > 0 \,\}$, with $\min \emptyset = \infty$.*

Proof. If $\mathrm{ov}_h(u,v) > 0$ there is $b \in \mathcal{N}(v)$ at distance exactly h from u, so $d(u,v) \leq h+1$. Conversely, if $d(u,v) = m < \infty$, the node before v on a shortest path (u itself when $m = 1$) lies at distance $m-1$ from u and is adjacent to v, so $\mathrm{ov}_{m-1}(u,v) > 0$; and $\mathrm{ov}_h(u,v) > 0$ for some $h < m-1$ would give $d(u,v) \leq h+1 < m$. □

Distance is therefore the depth at which the overlap first fires, and overlap can add to distance only the magnitude of its counts. The same argument bounds the other end of the overlap vector, and that bound is how a structural feature comes to carry memory.

Proposition 2. *For any graph and any $v \neq u$, $\mathrm{ov}_h(u,v) = 0$ whenever $h > d(u,v) + 1$. With Proposition 1, the overlap vector is supported on $\{d-1, d, d+1\}$, $d = d(u,v)$.*

Proof. Every $w \in \mathcal{N}(v)$ satisfies $d(u,w) \leq d(u,v) + 1$, so $w \notin \mathcal{N}_h(u)$ for $h > d(u,v) + 1$; the lower end is Proposition 1. □

Corollary 1. *If $\mathrm{ov}_h(u,v) > 0$ for some $h \geq 3$ then $d(u,v) \geq 2$: the pair has not interacted. A scorer that reads the overlap vector therefore reads whether the pair has ever met, which is memory rather than structure.*

The data agree exactly: across the 14 benchmarks and both protocols, no query whose pair had interacted before has a non-zero three-hop overlap (0 exceptions in 28 runs). The distinction matters because the two are easy to confuse. The number of past events of a pair enters a linear scorer as $\log(1 + \text{count})$, which cannot express "never" as a category; the overlap vector can, and does. After memory and recency, distance adds a median of +0.000 AP under random negatives and +0.000 under historical ones, and overlap adds +0.001 and +0.000 after all three.

6 WHERE STRUCTURE SURVIVES

Structure survives the test in two opposite situations, and in both of them memory cannot answer.

No memory to have. TGB-Seq (Yi et al., 2025) builds benchmarks without repeated edges, because the standard ones reward memorization. On its GoogleLocal stream (1.9 million events, bipartite, no repeated edge), 0.86 of the test queries have both endpoints known and the pair new. Memory reaches AP 0.692 and recency adds +0.024. Overlap adds +0.121, and it keeps +0.120 after recency. Only distance absorbs it: with distance in the model, overlap is left with +0.005. Inside the cold regime, overlap is worth +0.140 and distance +0.130, and by Proposition 1 the two are largely the same information. YouTube, the social stream of the same suite (3.3 million events), behaves the same way. Memory reaches 0.887; overlap adds +0.030, keeps +0.016 after recency, and keeps only +0.003 once distance is in the model (1 seed; one replay of that stream takes 35 hours). Where pairs

Table 4: The cold regime under random negatives: pairs that are new although both endpoints are known. Gains are after memory and recency, means of three seeds, on the benchmarks where the regime is at least 5% of test queries.

benchmark	cold share	memory+recency AP	overlap	classical	DyGFormer co-occ.	Hawkeye
Can. Parl.	0.47	0.674	−0.021	−0.006	−0.005	**+0.025**
Enron	0.10	0.495	**+0.020**	**+0.040**	**+0.078**	**+0.026**
Flights	0.12	0.728	**+0.119**	**+0.178**	**+0.150**	**+0.058**
LastFM	0.10	0.440	−0.007	+0.000	+0.003	**+0.025**
MOOC	0.49	0.644	−0.000	+0.000	+0.000	**+0.104**
Myket	0.70	0.794	+0.008	+0.000	+0.001	+0.005
Reddit	0.07	0.732	**+0.036**	+0.000	+0.000	+0.009
UCI	0.25	0.798	+0.004	−0.008	−0.001	+0.002
Wikipedia	0.09	0.735	**+0.019**	+0.000	+0.003	−0.014

never repeat, structure is what there is, and it acts through proximity rather than through counts (3 seeds on GoogleLocal).

Memory saturated. On UN Trade almost every test pair has traded before. Memory therefore marks nearly every candidate as a partner, and recency adds only +0.014. Overlap still adds +0.038 after recency under random negatives, and so do the classical heuristics and both published encodings (Table 2). What distinguishes partners there is how embedded the relation is, and shared-partner counts measure exactly that. Hawkeye's residuals under historical negatives sit in the same kind of place, on UN Vote and Can. Parl., dense sessional streams where the cohesion of the current window says which group is acting together now. Its residual on MOOC is of the other kind: under random negatives it lives in the cold regime (Table 4). The sessional streams are also where the two scorers disagree most. On Can. Parl. under random negatives the linear scorer sees +0.103 and the trees -0.058, so we make no claim there.

The cold part of an ordinary benchmark. The same pattern appears inside the standard benchmarks, in the queries whose pair is new although both endpoints are known. On the 9 benchmarks where this regime holds at least 5% of the test queries (Table 4), overlap still adds more than 0.01 AP after memory and recency on 4 of them, and Hawkeye's channel on 5. The medians over the nine are +0.008 and +0.025. DyGFormer's co-occurrence reaches +0.150 on Flights, and overlap +0.119. This regime is a small part of most benchmarks, between 7% and 25% on the finely timestamped ones, so a gain of this size is invisible in an aggregate score.

The regime that pays is a property of the benchmark. If structure pays in one regime only, it only has to be computed there. We multiply the structural columns by an indicator of one regime and choose the regime on a held-out half of the validation period; the indicator is a function of memory alone, so it costs nothing to evaluate. Take the 20 cells where a structural feature is worth more than 0.01 AP without gating. There the gate keeps a median of 98% of the gain while computing the feature on a median of 66% of queries (cell-level numbers in the supplementary material). The regime it picks differs by benchmark: 13 of the cells gate to repeated pairs and 5 to cold ones, the two ends described above. The gate fails on Can. Parl., where the validation and test periods fall in different parliamentary sessions and a choice made on one does not transfer to the other.

A protocol that cannot see it. The historical protocol is empty exactly where structure matters most. On a benchmark with no repeated edges, every positive is a pair that has never met and every historical negative is a pair that has. The memory indicator alone separates them perfectly: on GoogleLocal it reaches AP 1.000, and no feature can add anything. A benchmark built to remove memorization and a protocol built to defeat memory cannot be used together.

7 Related Work

Heuristics and repetition in temporal link prediction. EdgeBank predicts an edge if it has been seen before and is a strong baseline (Poursafaei et al., 2022). Daniluk & Dąbrowski (2023) show

that a recently-popular-nodes baseline outperforms temporal graph networks on the Temporal Graph Benchmark (Huang et al., 2023). Cornell et al. (2025) show that recency and popularity heuristics match or beat neural models on two benchmark suites. Yi et al. (2025) construct benchmarks with few repeated edges because existing ones reward memorization. These works show that heuristics compete with models; we ask whether the structural information models are built to extract is separable from memory and recency, for any feature.

Structure in temporal and static link predictors. Counting shared neighbors is among the oldest link predictors (Liben-Nowell & Kleinberg, 2007). SEAL labels an enclosing subgraph by distances to both endpoints (Zhang & Chen, 2018). Neo-GNNs learn a generalization of overlap over multi-hop neighborhoods (Yun et al., 2021). BUDDY estimates the number of nodes at each pair of distances from the two endpoints (Chamberlain et al., 2023); Proposition 1 reads off the first non-zero entry of that table. In temporal models, memory-based and attention-based predictors summarize a node's recent neighborhood (Rossi et al., 2020; Xu et al., 2020; Cong et al., 2023). Temporal walks sample paths between the endpoints (Wang et al., 2021). DyGFormer encodes neighbor co-occurrence (Yu et al., 2023). TNCN and TPNet encode temporal common neighbors and walk matrices (Zhang et al., 2024; Lu et al., 2024). HyperEvent encodes the relative structure of event sequences (Gao et al., 2025). Hawkeye encodes neighborhood cohesion in a sliding window (Ding & Zhang, 2026).

8 Limitations

The test reads features through two simple scorers and fixed-size summaries, and Section 4 checks its conclusion inside DyGFormer and TGN rather than inside every model. One negative is drawn per event, following DyGLib; the ranking protocol of TGB (Huang et al., 2023) may change absolute numbers. On the benchmarks recorded once a year the two scorers disagree most, and we draw no conclusion from those cells.

9 Conclusion

On the standard temporal link-prediction benchmarks, most of the gain credited to neighborhood structure is memory and time. A structural feature reveals whether a pair has ever met, by a property of the overlap vector that we prove, and it moves with how recently the endpoints were active. Once a predictor knows both, little is left, and what is left clears two scorers and a shuffled control in 14 of 112 cells. Structure pays where memory cannot answer: where no pair repeats, where every pair does, and in the cold part of an ordinary benchmark, which an aggregate score hides. Structural channels should therefore be evaluated where they can matter, and `structaudit` makes that evaluation cheap.

Reproducibility Statement

The test is released as `structaudit`: one command replays a benchmark and saves a report and the feature matrices, and every other step reads those matrices. The tool, the scripts that generate every table and figure, every per-run report and the DyGLib patch for the pair-time channel are in the supplementary material. All datasets are public.

Statement on the Use of Generative AI

We used a generative AI assistant to help develop the framework, formulate and prove the propositions, design and run the experiments, implement the methods, interpret the results, write code and figures, search the literature, and draft and edit the text. We did not use it to generate or clean data. The author reviewed all AI-assisted work and takes responsibility for the content.

REFERENCES

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *International Conference on Learning Representations*, 2023.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *ICLR*, 2023.

Filip Cornell, Oleg Smirnov, Gabriela Zarzar Gandler, and Lele Cao. On the power of heuristics in temporal graphs. *arXiv preprint arXiv:2502.04910*, 2025.

Michał Daniluk and Jacek Dąbrowski. Temporal graph models fail to capture global temporal dynamics. In *NeurIPS 2023 Workshop on Temporal Graph Learning*, 2023.

Jiacheng Ding and Xiaofei Zhang. HAWKEYE: Seeing one layer deeper – a cohesion-aware structural channel for temporal link prediction. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM)*, 2026. doi: 10.1145/3799682.3841168.

Jian Gao, Jianshe Wu, and JingYi Ding. HyperEvent: A strong baseline for dynamic link prediction via relative structural encoding. *arXiv preprint arXiv:2507.11836*, 2025.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. In *NeurIPS Datasets and Benchmarks*, 2023.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

Xiaodong Lu, Leilei Tian, Liyong Liu, et al. Improving temporal link prediction via temporal walk matrix projection. In *NeurIPS*, 2024.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. In *NeurIPS Datasets and Benchmarks*, 2022.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML Workshop on Graph Representation Learning*, 2020.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *ICLR*, 2021.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Yue Zixuan, and Zengfeng Huang. TGB-Seq benchmark: Challenging temporal GNNs with complex sequential dynamics. In *International Conference on Learning Representations*, 2025.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *NeurIPS*, 2023.

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. Neo-GNNs: Neighborhood overlap-aware graph neural networks for link prediction. In *Advances in Neural Information Processing Systems*, 2021.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, 2018.

Xiaohui Zhang, Yanbang Wang, Lijun Sun, and Pan Li. Efficient neural common neighbor for temporal graph link prediction. *arXiv preprint arXiv:2406.07926*, 2024.