Computers and Artificial Intelligence; Vol.3, No.6; 2026
E-ISSN: 3006-3728 P-ISSN: 3006-371X
DOI: https://doi.org/10.70267/cai.v3n6.152163
Published by: Zeus Press

Dynamic Graph Link Prediction by Integrating Historical Recurrence Signals and Time-Weighted Similarity Matching

Zhenhua Chen* and Run Zhong

School of Information, Guizhou University of Finance and Economics, Guiyang, Guizhou, China

**Corresponding author: Zhenhua Chen.*

Abstract

Future link prediction on dynamic graphs must rank both recurring interactions and links between previously unconnected nodes. We extend the CRAFT-R candidate-aware cross-attention matcher with explicit historical recurrence statistics, time-weighted embedding similarity, and candidate-specific residual fusion. Under chronological splits and fixed 100-candidate queries on three competition datasets, we evaluate four retrained configurations alongside history-based ranking rules. The full model achieves test mean reciprocal ranks (MRRs) of 0.845, 0.296, and 0.576 on dataset1, dataset2, and dataset3, respectively, exceeding the backbone control by 0.033, 0.026, and 0.033. Removing the statistics branch reduces MRR by 0.020, 0.007, and 0.004; removing the similarity branch reduces it by 0.012, 0.021, and 0.012. Separate strict-past analyses show that direct recurrence is sensitive to historical coverage and to whether recent events remain visible at prediction time. The auxiliary signals therefore contribute differently across datasets, while the value of a longer or rolling history depends on the available evidence. The full model also leads seven evaluated baselines on AP, AUC, and MRR across the three datasets under the reported protocol.

Keywords

dynamic graphs, future link prediction, repeated interactions, candidate-aware matching, mean reciprocal rank (MRR)

1. Introduction

Future link prediction uses past interactions to rank candidate destinations at a given time. The central difficulty is that an observed pair can recur, whereas a previously unconnected pair requires evidence that transfers beyond direct co-occurrence. We study both cases under one candidate-based ranking task and distinguish recurrence in the complete event stream, prior observation at query time, and evidence retained in a finite history window. We then evaluate two auxiliary neural scores, one based on direct recurrence and the other on embedding similarity, with matched component removals. Separate rule experiments identify how history availability affects ranking, so changes in input evidence are not confused with architectural gains.

© 2026 The Author(s). Authors retain copyright in their articles.

1.1 Research Background and Problem Statement

Temporal graphs describe the evolution of relationships through timestamped interaction events and can represent dynamic connections in recommendation, social, and other interaction systems. Given a source node, a query time, and a set of candidate destinations, future link prediction ranks candidates using observed events. For repeated interactions between previously connected node pairs, interaction frequency and the time of the most recent interaction may provide direct evidence. Previously unconnected node pairs lack this evidence and require transferable information from neighborhood structure, behavioral sequences, or node embeddings. A model must therefore preserve useful recurrence signals while distinguishing new node pairs.

Recurrence patterns vary across datasets and change the relative importance of these sources of evidence. Repeated directed node-pair events account for approximately 72.60%, 2.21%, and 32.12% of the complete training files of the three competition datasets, respectively. However, the global recurrence proportion does not equal the recurrence information available at prediction time. Events with the same timestamp as a query cannot be treated as past information, and older repeated interactions may fall outside the history window. Ignoring these conditions may cause a high overall score to reflect an advantage from observed past interactions while providing limited evidence of performance on new interactions. We therefore distinguish the global recurrence proportion, the proportion of node pairs observed strictly before the query time, and coverage within the recent history window, and include history visibility in the evaluation protocol.

The candidate-aware cross-attention model CRAFT [1] matches a destination with the source's interaction history, and its CRAFT-R variant also encodes repeat counts. We retain that inherited matcher and test whether explicitly computed temporal recurrence statistics and a time-weighted embedding match add evidence beyond those representations. The resulting residual formulation is an incremental model extension; strict-past coverage, tie handling, and history visibility are separate empirical analyses. Retrained component variants use matched candidate sets and training objectives so that branch effects can be assessed directly.

1.2 Related Work

Temporal graph learning captures dynamic dependencies through node states, temporal neighborhoods, and structural patterns. JODIE uses coupled recurrent networks to update the dynamic embeddings of interacting nodes [2], while TGN combines node memory with graph operators to model interactions in continuous time [3]. TGAT uses time encoding and neighborhood attention for inductive representation learning [4], and CAW extracts transferable temporal structural patterns through causal anonymous walks [5]. GraphMixer processes temporal information using multilayer perceptrons and neighborhood aggregation [6], whereas DyGFormer uses neighbor co-occurrence encoding and sequence patching to model node relationships and longer interaction histories [7]. These methods show that the organization of historical interactions and the types of available information should be considered when evaluating dynamic graph models.

Sequential recommendation and research on repeated interactions provide further motivation for history matching. GRU4Rec uses a recurrent network to model behavioral transitions within a session [8], SASRec selects relevant historical behaviors through self-attention [9], and Transformer provides a foundational architecture for attention-based sequence modeling [10]. RepeatNet explicitly distinguishes between repeat and exploration modes [11]. The EdgeBank baseline shows that prediction on some temporal graph datasets can rely heavily on previously observed interactions [12]. LightGCN learns collaborative representations through embedding propagation on a graph [13]. Our similarity score instead compares candidates directly with historical-neighbor embeddings and does not implement LightGCN's propagation. These studies motivate separate measurement of direct recurrence and embedding-based matching rather than treating them as interchangeable explanations.

Training objectives and evaluation protocols also affect comparisons between methods. BPR learns rankings through pairwise comparisons of positive and negative samples [14], while Focal Loss adjusts sample weights according to prediction difficulty [15]. Our method adopts a reweighting mechanism but stops gradients through the weights; its optimization process therefore differs from that of the original Focal Loss. TGB emphasizes consistent evaluation across datasets and tasks [16], and TGB-Seq further examines complex sequential dynamics and prediction for previously unconnected node pairs [17]. Research on sampled ranking metrics also shows that conclusions based on a subset of candidates may not hold for full-catalog ranking [18]. Accordingly, we use fixed candidate sets, specify how tied scores are ranked, and distinguish between frozen

© 2026 The Author(s). Authors retain copyright in their articles.

and rolling history. Rule-based results, the original binary classification validation results, and neural component comparisons are interpreted under their respective evaluation protocols.

1.3 Research Approach and Main Contributions

Our incremental extension adds two explicit scores to the CRAFT-R candidate-aware matcher. The recurrence branch summarizes candidate occurrence, frequency, recency, and a temporal density proxy; the similarity branch pools cosine similarity and dot products against historical-neighbor embeddings with time-dependent weights. Candidate-ID-specific coefficients and a destination bias combine these scores with the backbone output. A margin-based pairwise loss with detached difficulty weights trains the combined score. These design choices are implemented in the accompanying code, but the coefficients are unnormalized and should not be interpreted as probabilities of selecting distinct experts.

The empirical contribution combines component-level testing with a controlled analysis of what history-based ranking can use. We evaluate four neural configurations under chronological, strict-past queries with fixed 100-candidate sets, then examine observed-pair coverage, score ties, history length, and frozen versus rolling visibility on three anonymized competition datasets. The component experiment isolates each residual branch; the rule experiments identify how much direct historical information was available to any scorer. Chapter 2 states testable hypotheses, Chapter 3 defines the scoring model, Chapter 4 reports these results and a seven-model external comparison, and Chapter 5 discusses the supported conclusions and remaining validation needs.

2. Research Hypotheses

To connect dataset characteristics, model mechanisms, and experimental interpretation, we formulate hypotheses about the usefulness of historical evidence and the complementarity of the two added scores. The history hypotheses are examined through controlled rule comparisons, while the component hypothesis is assessed using four retrained neural configurations. These forms of evidence are reported separately because changing the available history is different from changing the model architecture.

2.1 Historical Recurrence Signals and History Visibility

A rule based on repeated interactions can directly assign a matching score to a candidate only when its past connection with the source node is visible. Repeated events in the complete event stream may share the query timestamp or fall outside the current history window; their global proportion therefore cannot directly predict the rule's ranking performance. Even when the positive candidate has appeared before, historical negative candidates may also receive high scores, and the final rank depends on their relative evidence within the candidate set. We therefore propose H1: given identical queries, candidates, and treatment of tied scores, the ranking performance of recurrence-based rules is related to the coverage of positive candidates in the history strictly preceding the query time. Datasets with higher coverage are more likely to provide useful distinctions based on frequency and recency. We examine this hypothesis using overall rule-based MRR and results grouped by previously connected and previously unconnected node pairs.

History length and history visibility determine whether the model can access this recurrence evidence. A longer window can recover earlier interactions, while rolling history can incorporate events that occurred after the training cutoff but before the current query. Either change may increase coverage of positive candidates, but it may also raise the scores of negative candidates. We therefore propose H2: when expanding the history window or updating rolling history increases useful recurrence evidence for positive candidates, the MRR of the recency rule is expected to improve; without such an increase in coverage, a consistent gain is not expected. The history-window experiment holds history visibility fixed, whereas the frozen-versus-rolling experiment holds window length fixed. Both use paired queries so that changes in available information are not attributed to changes in model architecture.

2.2 Complementarity of the Statistics and Similarity Branches

The inherited matcher already uses repeat-count encoding and cross-attention to read interaction history. Explicit recurrence statistics therefore supply a transparent residual signal rather than introduce interaction

© 2026 The Author(s). Authors retain copyright in their articles.

memory for the first time. The similarity branch compares a candidate embedding with historical-neighbor embeddings and can produce a nonzero score without a direct prior connection. Time weighting emphasizes recent events. The two branches may be complementary, but they are not exclusive repeat and exploration experts: similarity can also score repeated candidates, and recurrence statistics may overlap with information already encoded by the backbone. The candidate-ID-specific coefficients adjust their contributions, but their usefulness for training-unseen destinations is an empirical question.

We therefore state H3: under the same backbone, destination bias, training objective, and evaluation protocol, the two auxiliary scores provide complementary information and improve ranking relative to variants with either or both branches removed. We test this prediction with four configurations under common chronological splits, queries, and candidate sets. A larger statistics contribution on previously connected pairs and a larger similarity contribution on previously unconnected pairs would support the proposed mechanisms, although the available aggregate results do not yet resolve those group-specific effects. Rule-based MRR and earlier AP/AUC records are kept separate from this neural ablation.

3. Research Design

This chapter defines the implemented scoring process from input history through the final candidate ranking. The existing candidate-aware cross-attention matcher produces a backbone score from interactions before the query time. Two auxiliary branches compute explicit recurrence statistics and time-weighted embedding similarity from the same history, and candidate-specific coefficients combine their outputs as residual scores. The pairwise objective is then specified separately from the evaluation protocol so that training choices and history visibility are not conflated.

3.1 Prediction Task and Backbone Representation

An interaction event is denoted by (u, v, t), where u is the source node, v is the destination node, and t is the interaction time. Given u, t, and a candidate set C, the model assigns a real-valued score s(u, v, t) to each $v \in C$ and ranks candidates in descending score order. Let $H_u(t)$ contain the L most recent events involving u with timestamps strictly earlier than t. For its j-th event, v_j denotes the other node and t_j its timestamp. The number of available historical events is n, where $0 \leq n \leq L$. Repeated interactions remain separate events. The implementation pads shorter histories to L positions, but padded positions are excluded from the statistics and similarity aggregation below. A previously unconnected node pair is one that has not interacted before the query time; neither node necessarily has to be new.

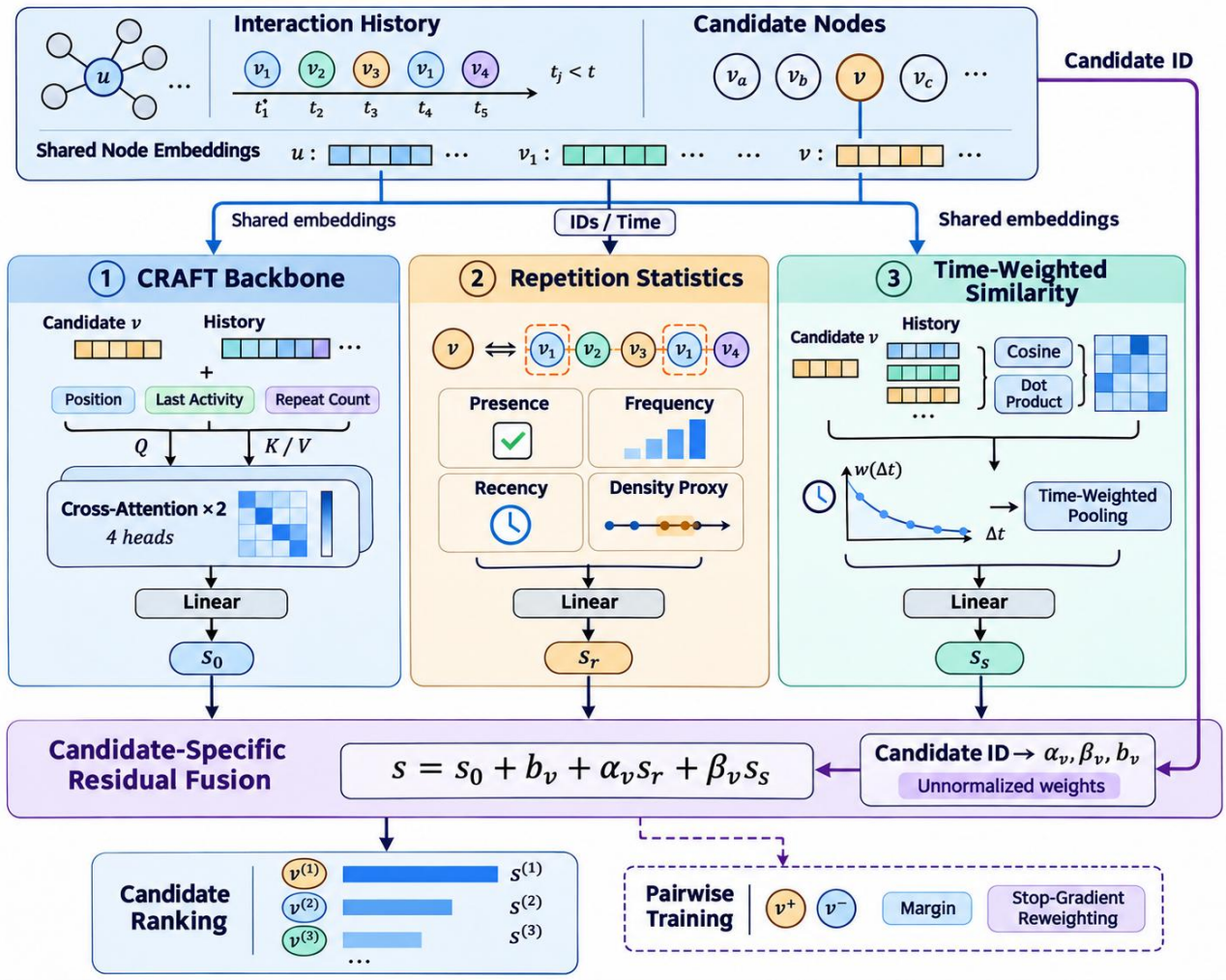
The cross-attention backbone follows the CRAFT-R variant: each candidate destination serves as a query over the source node's history. Positional, destination last-activity, and repeat-count encodings remain in place. We denote its output by:

$$s_0 = F(u, v, t, H_u(t)) \tag{1}$$

In Eq. (1), s_0 is the inherited backbone score for the current candidate, and F is its trainable matching function. The arguments u, v, t, and $H_u(t)$ denote the source node, candidate destination, query time, and interaction history, respectively. The implementation uses 128-dimensional shared node embeddings and two cross-attention layers with four heads. Neighbor history is bidirectional. Rolling history includes events observed before a query even when they occur after the training cutoff, whereas frozen history uses only training-period events. Both exclude the query timestamp and future events; model parameters remain fixed during evaluation. Figure 1 places the two scoring branches added in this study alongside the inherited CRAFT-R backbone.

© 2026 The Author(s). Authors retain copyright in their articles.

Figure 1: Architecture of the proposed dynamic graph link prediction model



3.2 Historical recurrence statistics and time-weighted embedding similarity

The historical recurrence statistics branch converts candidate occurrence, frequency, recency, and a temporal density proxy into an auxiliary score. Let I_j equal 1 when $v_j = v$ and 0 otherwise. For each valid historical event, $\ell_j = \ln(t - t_j + 2)$ is the log time interval, where ln denotes the natural logarithm. Frequency f is the sum of I_j, and occurrence indicator z equals 1 if $f > 0$ and 0 otherwise. Recency r is the largest $1/\ell_j$ among matching events, or 0 if there is no match. We further define $a = (\Sigma_j I_j \ell_j)/(f + \varepsilon)$ and $d = f/(a + \varepsilon)$, where d is a temporal density proxy and $\varepsilon = 10^{-8}$ prevents division by zero. A linear combination of the four statistics gives:

$$s_r = w_{r,z} z + w_{r,f} f + w_{r,r} r + w_{r,d} d \tag{2}$$

In Eq. (2), s_r is the explicit recurrence score. The four trainable scalars w_r,z, w_r,f, w_r,r, and w_r,d multiply z, f, r, and d, respectively; this branch has no separate bias. Larger r indicates a more recent matching interaction. The density proxy d depends on frequency and time intervals but is neither an interaction probability nor a point-process intensity. If the candidate is absent from the history window, $z = f = r = d = 0$ and thus $s_r = 0$. The pre-existing repeat-count encoding remains in the backbone, so this branch tests the added value of a direct statistics-based residual.

The time-weighted embedding similarity branch uses the same node embeddings. Let e_v be the embedding of candidate v and e_j the embedding of historical neighbor v_j. Set $g_j = 1/\ell_j$ and $\rho_j = g_j/\max(\Sigma_k g_k, \varepsilon)$, with the sum taken only over the n valid historical events; k is a summation index. The time-aggregation coefficient ρ_j generally assigns greater weight to more recent events. It is calculated from time intervals rather than learned as a candidate-specific residual weight. These coefficients are used to compute two matching measures:

$$c = \sum_{j=1\ldots n} \rho_j \cos\varepsilon(e_v, e_j) \; q = \sum_{j=1\ldots n} \rho_j e_v{}^T e_j \tag{3}$$

© 2026 The Author(s). Authors retain copyright in their articles.

In Eq. (3), c is the time-weighted cosine similarity and q is the time-weighted dot product; superscript T denotes vector transpose. The function cosε(x, y) normalizes x and y by max(‖x‖₂, ε) and max(‖y‖₂, ε), respectively, before taking their dot product, where ‖·‖₂ is the Euclidean norm. This convention also defines the calculation for zero vectors. If the history is empty, both sums equal zero and c = q = 0. The two measures form the similarity score:

$$s_s = w_{s,c}c + w_{s,q}q \tag{4}$$

In Eq. (4), s_s is the similarity branch score, while w_s,c and w_s,q are trainable scalars for the cosine and dot-product terms; this branch has no separate bias. Cosine similarity emphasizes vector direction, whereas the dot product also depends on vector magnitude. A candidate may have high similarity to historical neighbors even if it has not interacted directly with the source node; repeated candidates can also receive similarity scores. Thus, the two added branches provide different types of evidence but are not constrained to act as mutually exclusive experts.

3.3 Residual Scoring and Pairwise Training

For a given query and candidate, the three scores are combined as shown in Eq. (5). Their common arguments (u, v, t) are omitted for brevity.

$$s = s_0 + b_v + \alpha_v s_r + \beta_v s_s \tag{5}$$

In Eq. (5), s is the final ranking score, and s_0, s_r, and s_s are the backbone, historical recurrence statistics, and similarity scores, respectively. The trainable destination bias b_v belongs to candidate v, while α_v and β_v are its two residual coefficients. Each coefficient comes from a scalar embedding indexed by the candidate ID. During training, dropout with a rate of 0.2 is applied to the raw coefficients, which are then divided by a shared trainable scaling value clamped to a minimum of 0.1. The coefficients are not passed through softmax: they may be negative and need not sum to one. Because they depend only on the candidate ID, they are not interpreted as query-dependent expert selection probabilities.

Each positive training example is paired with one uniformly sampled negative example. Let s_i^+ and s_i^- be their scores for pair i, and let $\delta_i = s_i^+ - s_i^- - m$ be the score difference after subtracting margin m = 0.5. We then calculate $p_i = \sigma(\delta_i)$, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function and exp is the exponential function. A larger p_i indicates that the positive example is ranked more favorably relative to its negative counterpart. The sample weight is $\omega_i = \text{stopgrad}[(1 - p_i)^2]$, where stopgrad means that gradients do not flow through the weight calculation. The batch loss is:

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1\ldots B}\omega_i \, ln(p_i + \varepsilon) \tag{6}$$

Here, $\mathcal{L}$ is the training loss to be minimized, B is the number of example pairs in a batch, i indexes those pairs, and ω_i places relatively more weight on difficult comparisons. The numerical stability constant ε remains 10^{-8}. The objective operates on pairwise comparisons; an individual candidate score is not interpreted as a calibrated probability. Evaluation ranks candidates using their raw scores. For K candidates per query, a history length L, and an embedding dimension d_e, the added similarity computation has leading time complexity $O(BKLd_e)$ and requires O(BKL) space for the similarity matrix. Here K is the number of candidates and d_e is the embedding dimension, distinct from candidate set C and density feature d.

4. Empirical Analysis

The empirical analysis combines measured neural ablations with history-based ranking diagnostics under a fixed offline protocol. We first specify datasets, temporal splits, candidate construction, and tie handling. Section 4.2 consolidates the available validation and test records, compares seven measured external baselines, and analyzes observed ranking behavior. History-length and visibility experiments identify changes in accessible evidence, while the four-configuration ablation estimates the added value of each residual score.

4.1 Datasets and Experimental Setup

We analyze only dataset1, dataset2, and dataset3. Their sizes and historical coverage are reported in Tables 1 and 2. The recurrence ratio is computed as one minus the number of distinct directed node pairs divided by

© 2026 The Author(s). Authors retain copyright in their articles.

the number of events. It includes repeated events at the same timestamp and therefore does not equal the amount of past information available at prediction time. The historically seen-pair rate and coverage by the 100 most recent events are calculated from strictly prior events under bidirectional rolling history. A training-unseen destination is an ID that did not occur as a destination in the training segment. Source and destination ID sets overlap in dataset1 and dataset3 but are disjoint in dataset2; we do not infer application semantics from these ID relationships.

Table 1: Dataset sizes and repeated-event ratios for directed node pairs

Dataset	Events	Source nodes	Destination nodes	Unique nodes	Recurrence ratio
dataset1	690848	22093	23012	23852	72.60%
dataset2	2261283	12708	50640	63348	2.21%
dataset3	1042895	36223	43945	44076	32.12%

Table 2: Historical coverage in the fixed offline test queries

Dataset	Queries	Historically seen pairs	Coverage in latest 100	Training-unseen destinations
dataset1	10000	82.58%	81.14%	16.48%
dataset2	10000	0.00%	0.00%	19.75%
dataset3	10000	60.30%	43.95%	0.28%

The complete timestamp groups are divided approximately 70%, 15%, and 15% into training, validation, and test segments, respectively; no timestamp group crosses a split boundary. From each of the validation and test segments, 10,000 events are sampled uniformly without replacement and sorted by time. The candidate seed is 2026, with an independent offset seed for the test set. Each query contains one positive and 99 negatives sampled uniformly without replacement from training-period destination IDs, and the positive is placed at a random position. All labeled positives for the same source and timestamp are excluded from the negative candidates. Previously interacted destinations may serve as negatives, and future labels are not used for filtering. A positive destination may be unseen during training. Thus, this fixed 100-candidate protocol differs from full-catalog ranking and the official candidate distribution [18].

The primary metric is mean reciprocal rank (MRR). Tied scores are assigned their mean rank, as defined in Eq. (7). Here, N is the number of queries and $rank_i$ is the rank of the positive candidate in query i; G_i counts candidates with strictly higher scores, and T_i counts candidates tied with the positive, including the positive itself. All methods use the same queries and candidates. Neural models are ranked by their raw scores to limit ties caused by output rounding; Section 4.2 also examines the effect of random tie-breaking on rule-based results.

$$MRR = \frac{1}{N}\sum_{i=1\ldots N}\frac{1}{rank_i}\ rank_i = 1 + G_i + \frac{T_i-1}{2} \tag{7}$$

The rule-based baselines comprise global destination frequency in the training segment, repeat count within the history window, the reciprocal log time interval of the most recent match, and the sum of reciprocal log time intervals over all matches. They diagnose historical signals but do not replace the neural-component ablations. The prespecified main configuration uses L = 100 and rolling history. The comparisons across $L \in \{20, 50, 100, 200\}$ and frozen history are sensitivity analyses; the main configuration is not reselected using test results.

Neural retraining retains 128-dimensional embeddings and a two-layer, four-head backbone. Backbone dropout is 0.15, and fusion-weight dropout is 0.2 [19]. The Adam optimizer [20] uses an initial learning rate of 10^{-4}, weight decay of 10^{-5}, and a batch size of 200. Training runs for at most 100 epochs, with early stopping after 15 epochs without improvement and learning-rate reduction after four such epochs. The original full-model run selects its checkpoint by validation AP, while the four retrained component variants use validation MRR; test results are not used for checkpoint selection. Fixed candidates and their metadata, per-query rule-based rankings, and neural training records are handled separately. The test comparison of the four neural configurations is reported in Section 4.4.

4.2 Prediction Results and Mechanism Analysis

Table 3 compares seven temporal-graph baselines with the full model on three datasets. AP and AUC are calculated from binary validation records, whereas MRR is calculated from the fixed 100-candidate offline

© 2026 The Author(s). Authors retain copyright in their articles.

test queries described in Section 4.1. These metrics describe different evaluation tasks and should be compared only within the same metric and dataset.

Table 3: Validation AP and AUC and offline test MRR across three datasets

Model	dataset1			dataset2			dataset3		
	AP	AUC	MRR	AP	AUC	MRR	AP	AUC	MRR
EdgeBank	0.904	0.906	0.812	0.875	0.885	0.188	0.801	0.798	0.478
JODIE	0.945	0.948	0.802	0.925	0.924	0.197	0.855	0.841	0.488
TGN	0.956	0.958	0.823	0.926	0.921	0.204	0.877	0.871	0.493
TGAT	0.961	0.956	0.825	0.923	0.927	0.208	0.881	0.859	0.485
CAW	0.960	0.962	0.826	0.918	0.925	0.256	0.885	0.872	0.516
GraphMixer	0.971	0.970	0.834	0.940	0.937	0.243	0.895	0.889	0.528
DyGFormer	0.975	0.969	0.838	0.941	0.940	0.278	0.901	0.888	0.539
Full model	**0.981**	**0.976**	**0.845**	**0.945**	**0.946**	**0.296**	**0.908**	**0.892**	**0.576**

Note: AP and AUC are validation metrics; MRR is measured on the offline test set. All values are rounded to three decimal places, and higher is better.

The full model ranks first on all six validation metrics. Its AP exceeds the strongest baseline, DyGFormer, by 0.006, 0.004, and 0.007 on dataset1, dataset2, and dataset3, respectively. The corresponding AUC margins over the strongest baselines are 0.006, 0.006, and 0.003; GraphMixer is the closest on dataset1 and dataset3, while DyGFormer is the closest on dataset2. These small differences should be interpreted as observed performance gaps rather than evidence of statistical significance.

On the offline ranking task, the full model reaches MRRs of 0.845, 0.296, and 0.576. DyGFormer is the strongest external baseline for MRR on each dataset, scoring 0.838, 0.278, and 0.539. The full-model margins are therefore 0.007, 0.018, and 0.037 MRR. The largest absolute ranking gain occurs on dataset3. Dataset2 yields lower MRR for every method, although its validation AP and AUC remain high; the validation and ranking metrics use different candidate settings and their numerical scales are not directly comparable.

Historical visibility helps interpret these differences. Table 2 shows that 82.58% and 60.30% of test positives in dataset1 and dataset3, respectively, involve node pairs observed strictly before the query, whereas dataset2 has no such positive pairs in the sampled queries. The most-recent-interaction rule obtains MRRs of 0.811333, 0.019771, and 0.442619 under the same offline queries. This contrast indicates that direct recurrence is useful when prior pair interactions are visible, but it does not establish which learned component drives the full-model advantage. The matched branch removals in Section 4.4 provide that component-level test.

The rule analysis also shows why ranking conventions matter: on dataset2, its mean-rank MRR of 0.019771 becomes 0.050412 under expected random tie-breaking because many candidates receive identical zero scores. The neural models are evaluated from unrounded scores, but the available aggregate comparison does not include multiple training seeds or paired per-query uncertainty for all baselines. Consequently, Table 3 supports a descriptive performance comparison under the reported protocol, not a significance claim or a general ranking of the methods across other temporal splits and candidate sets.

4.3 History Length and Visibility Analysis

To examine the window effect posited by H2, we keep rolling history and test candidates fixed while setting history length to 20, 50, 100, or 200. Figure 2 shows how MRR changes with the window length, and positive-event window coverage helps explain the changes.

When the window increases from 100 to 200 in dataset1, coverage rises from 81.14% to 82.10% and MRR increases by 0.008860, with a paired interval of [0.007101, 0.010724]. For dataset3, coverage rises from 43.95% to 53.70% and MRR increases by 0.084727, with an interval of [0.078982, 0.090564]. The latter result indicates that truncation omits many usable older interactions. Whether a longer L is worthwhile for a neural model also depends on attention and similarity-branch costs and validation performance. In dataset2, positive-event coverage is zero at every window length. Its MRR declines slightly as the window grows, possibly because additional history gives more negatives nonzero scores rather than improving positive-event memory. More history alone therefore does not guarantee a gain.

Figure 2 displays the history-window results on separate vertical scales to make each dataset's trend visible; numerical differences must therefore be compared from the stated MRR values rather than line steepness.

© 2026 The Author(s). Authors retain copyright in their articles.

Dataset1 and dataset3 improve as more past interactions are admitted, in line with their rising positive-event coverage. Dataset2 remains close to the all-tied baseline and declines slightly as the window expands. All plotted values concern the most-recent-interaction rule, not the neural model.

Figure 2: Recency-rule MRR across history windows with independent vertical scales



At L = 100, we further compare frozen history, which uses only training-period events, with rolling history, which also includes events observed before each prediction time. Figure 3 shows the most-recent-interaction rule under these two information conditions.

Relative to frozen history, rolling history improves MRR by 0.385286 on dataset1 and 0.171614 on dataset3, with paired intervals of [0.371820, 0.397943] and [0.161566, 0.182313], respectively. Rolling history can use interactions that occur after the training cutoff but before the prediction time, increasing historical coverage. The difference arises from a change in available information and must not be interpreted as an architectural improvement. The difference on dataset2 is close to zero, consistent with its lack of usable strictly prior repeated positives in this analysis.

Figure 3: Recency-rule MRR under frozen and rolling history at L = 100

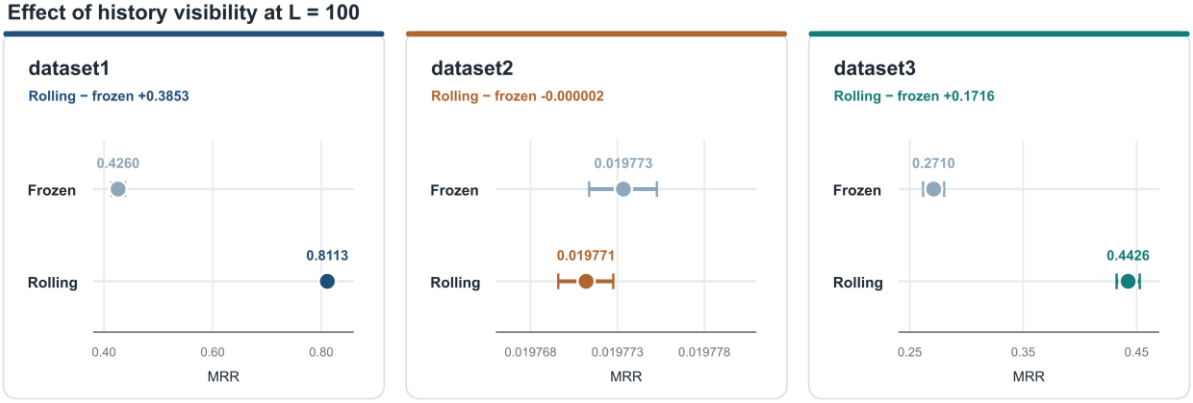


Figure 3 uses separate horizontal scales for the three datasets and shows conditional 95% source-node clustered bootstrap intervals. Rolling history markedly improves the rule on dataset1 and dataset3 by adding strictly prior events after the training cutoff. The two estimates on dataset2 nearly coincide. The comparison changes the observable evidence while leaving the scoring rule fixed; it cannot be read as an architectural gain.

4.4 Ablation Study

Table 4 compares the full model, each single-branch removal, and the inherited backbone with both auxiliary branches removed. The four variants use the same 100-candidate ranking task and reported chronological protocol. The backbone retains its original repeat-count encoding when the explicit statistics branch is removed; thus, that contrast isolates the added residual score rather than all use of repeated

© 2026 The Author(s). Authors retain copyright in their articles.

interactions. The MRR values come from the later retrained comparison and are reported to three decimal places.

Table 4: Test MRR of four matched ablation configurations

Model variant	Statistics branch	Similarity branch	dataset1 MRR	dataset2 MRR	dataset3 MRR
Full model	Included	Included	0.845	0.296	0.576
Without statistics	Removed	Included	0.825	0.289	0.572
Without similarity	Included	Removed	0.833	0.275	0.564
Backbone control	Removed	Removed	0.812	0.270	0.543

The full model reaches MRRs of 0.845, 0.296, and 0.576 on dataset1, dataset2, and dataset3. Relative to the backbone control, the gains are 0.033, 0.026, and 0.033 MRR, respectively. With similarity retained, removing explicit recurrence statistics lowers MRR by 0.020, 0.007, and 0.004. With statistics retained, removing time-weighted similarity lowers MRR by 0.012, 0.021, and 0.012. Both residual scores therefore have positive marginal contributions in all three reported datasets, but their magnitudes differ: the statistics branch contributes more on dataset1, whereas the similarity branch contributes more on dataset2 and dataset3. These paired model contrasts are distinct from the rule-based changes caused by lengthening or updating history.

The pattern is consistent with the history diagnostics without proving a unique mechanism. Dataset2 has no previously observed positive pairs in the sampled test queries, making its larger similarity contrast plausible; dataset1 has high strict-past pair coverage, making an added recurrence signal potentially useful. Dataset3 has intermediate coverage and benefits from both branches. Because the available ablation record contains aggregate MRR rounded to three decimals, it does not support confidence intervals, run-to-run standard deviations, or a claim about which node-pair subgroup drives each neural gain. The results support H3 at the overall-MRR level for these three datasets, while the proposed seen-pair versus unseen-pair explanation remains to be tested with per-query neural scores.

5. Conclusion

This study evaluates a residual scoring extension that combines explicit historical recurrence statistics, time-weighted embedding similarity, and candidate-specific coefficients with an inherited candidate-aware matcher. Under strict chronological splits and fixed 100-candidate test queries, the full model attains MRRs of 0.845, 0.296, and 0.576 on dataset1, dataset2, and dataset3, exceeding the matched backbone control by 0.033, 0.026, and 0.033. Removing either added branch lowers overall MRR on each dataset, with dataset-dependent effect sizes.

The history analysis distinguishes global recurrence, query-time pair visibility, and finite-window coverage. Strong aggregate rule scores can be dominated by previously connected pairs; longer windows recover useful old interactions in some datasets, while rolling history adds events that become visible after the training cutoff. The ablation results establish positive overall-MRR contributions for the two residual branches under the reported protocol. Their different effect sizes across datasets are compatible with the observed coverage patterns, but per-query neural results are needed to attribute gains to seen or unseen pairs.

The evidence remains limited in precision and scope. The available ablation and external-baseline results are aggregate values from the reported protocol, without repeated-seed variation or paired per-query scores for the neural methods; statistical significance of the observed model gaps therefore cannot be assessed. The rule comparisons use one temporal split and one candidate sample, and their source-node bootstrap omits training and candidate-resampling uncertainty. ID-indexed fusion weights may also limit performance for destinations unseen during training. These limitations restrict claims of generalization beyond the three competition datasets.

Future work should rerun the full model, component variants, and seven external baselines with multiple seeds and retain per-query scores for paired uncertainty analysis. A fuller report should include overall and seen-pair versus unseen-pair MRR, parameter count, peak memory, training time, and inference throughput. Additional temporal splits, candidate resamples, and public datasets would test whether the observed history effects and branch contributions generalize beyond these competition datasets.

© 2026 The Author(s). Authors retain copyright in their articles.

References

[1] Yi L, Lei R, Mo F, et al. Future Link Prediction Without Memory or Aggregation. Advances in Neural Information Processing Systems. 2025; 38.

[2] Kumar S, Zhang X, Leskovec J. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 1269–1278.

[3] Rossi E, Chamberlain B, Frasca F, et al. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In: ICML Workshop on Graph Representation Learning and Beyond. 2020.

[4] Xu D, Ruan C, Korpeoglu E, et al. Inductive Representation Learning on Temporal Graphs. In: International Conference on Learning Representations. 2020.

[5] Wang Y, Chang Y Y, Liu Y, et al. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In: International Conference on Learning Representations. 2021.

[6] Cong W, Zhang S, Kang J, et al. Do We Really Need Complicated Model Architectures for Temporal Networks? In: International Conference on Learning Representations. 2023.

[7] Yu L, Sun L, Du B, et al. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. Advances in Neural Information Processing Systems. 2023; 36: 67686–67700.

[8] Hidasi B, Karatzoglou A, Baltrunas L, et al. Session-Based Recommendations with Recurrent Neural Networks. In: International Conference on Learning Representations. 2016.

[9] Kang W C, McAuley J. Self-Attentive Sequential Recommendation. In: Proceedings of the IEEE International Conference on Data Mining. 2018: 197–206.

[10] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017; 30: 5998–6008.

[11] Ren P, Chen Z, Li J, et al. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-Based Recommendation. Proceedings of the AAAI Conference on Artificial Intelligence. 2019; 33(1): 4806–4813.

[12] Poursafaei F, Huang S, Pelrine K, et al. Towards Better Evaluation for Dynamic Link Prediction. Advances in Neural Information Processing Systems. 2022; 35: 32928–32941.

[13] He X, Deng K, Wang X, et al. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020: 639–648.

[14]Rendle S, Freudenthaler C, Gantner Z, et al. BPR: Bayesian Personalized Ranking from Implicit Feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence. Proceedings of Machine Learning Research. 2009; R7: 460–469.

[15] Lin T Y, Goyal P, Girshick R, et al. Focal Loss for Dense Object Detection. In: Proceedings of the IEEE International Conference on Computer Vision. 2017: 2980–2988.

[16] Huang S, Poursafaei F, Danovitch J, et al. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. Advances in Neural Information Processing Systems. 2023; 36.

[17] Yi L, Peng J, Zheng Y, et al. TGB-Seq Benchmark: Challenging Temporal GNNs with Complex Sequential Dynamics. In: International Conference on Learning Representations. 2025.

[18] Krichene W, Rendle S. On Sampled Metrics for Item Recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 1748–1757.

[19] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014; 15(56): 1929–1958.

[20] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations. 2015.

© 2026 The Author(s). Authors retain copyright in their articles.

Funding

This research received no external funding.

Conflicts of Interest

The author(s) declare no conflicts of interest.

Acknowledgment

This paper is an output of the science project.

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal. This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

© 2026 The Author(s). Authors retain copyright in their articles.