DO TEMPORAL LINK PREDICTORS NEED LEARNED MEMORY? A SMOOTHED-COUNT BASELINE WITH A HANDFUL OF PARAMETERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many temporal link predictors summarize past interactions through learned node representations. We examine whether simple counts of recurring interaction patterns can provide competitive predictions without learning these representations. We propose a temporal link predictor based on statistical language modelling. It pools transition and co-occurrence counts across sources to predict links that a source has never formed. We smooth sparse estimates using destination frequencies or Kneser-Ney continuation counts. A shared log-linear rule combines these estimates with popularity, source history, and recency, without node embeddings. In our main evaluation, the model achieves the highest MRR among the compared methods on 7 out of 16 datasets from TGB and TGB-Seq. It also outperforms EdgeBank and Base3 on all 16 datasets and the heuristic family on 14. These gains extend to datasets designed to limit repeated edges. With only 9–13 learned parameters, our model provides a simple and competitive baseline for evaluating future neural temporal link predictors.

1 INTRODUCTION

Temporal link prediction models learn from past interactions to predict future links. Many methods represent this history through learned node states, embeddings, or neighborhood encodings (Kumar et al., 2019; Trivedi et al., 2019; Rossi et al., 2020; Wang et al., 2021; Yu et al., 2023; Lu et al., 2024). Simple temporal patterns also carry predictive information. Edges repeat, and destination frequencies change over time (Poursafaei et al., 2022; Daniluk & Dąbrowski, 2023). Recent work therefore questions whether complex architectures are needed for temporal prediction (Cong et al., 2023). Strong statistical baselines help measure the predictive gains from learned representations.

Existing count-based methods use only part of this information. EdgeBank predicts the recurrence of previously observed edges (Poursafaei et al., 2022). PopTrack ranks destinations by decayed frequency (Daniluk & Dąbrowski, 2023). Other methods combine recency, popularity, and co-occurrence through fixed rules (Cornell et al., 2025a; Kondrup, 2025). These methods have mainly been studied on datasets with repeated edges. Their predictive value when sources rarely revisit a destination remains less understood. TGB-Seq tests this setting (Yi et al., 2025).

Temporal interactions provide recurring prediction contexts. The recent destinations of a source can inform its next destination, but individual histories often contain few observations of each transition. Statistical language modelling addresses this problem by pooling observations of recurring contexts and smoothing sparse estimates toward a broader distribution (Katz, 1987; Chen & Goodman, 1999).

We use this construction to build a temporal link predictor from smoothed counts and recency features, with a shared log-linear scoring rule. We consider two backoff distributions for contexts with few observations. Marginal backoff uses destination frequency, favoring items that are common overall. This follows the use of unigram frequencies in language-model interpolation (MacKay & Peto, 1995). Kneser-Ney continuation backoff uses the number of distinct preceding destinations, favoring items observed across many transition contexts (Kneser & Ney, 1995).

A destination can receive support from several statistics (Figure 1). It may be popular overall, frequent in the history of the source, or recently visited. It may also commonly follow a recent

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.

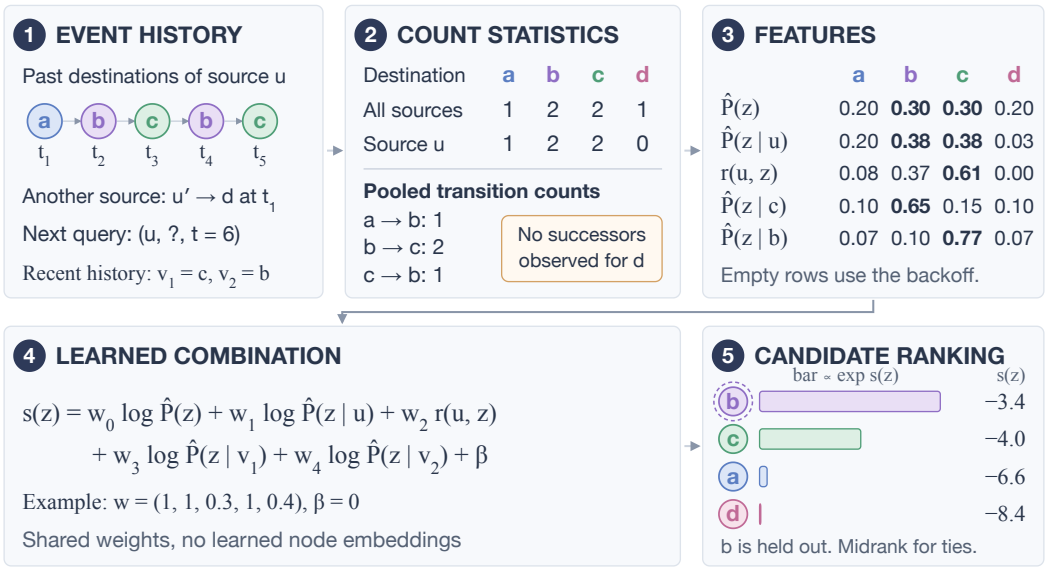

Figure 1: From six observed interactions to candidate ranking. The reference configuration uses $K = 2$, $\alpha = 1$, undecayed counts, and marginal backoff. We set $t_i = i$, query at $t = 6$, and use recency scale $\tau_r T = 2$. Panels 1–3 form the features. Panels 4–5 apply illustrative weights and rank candidates. Appendix E gives the calculations and further features.

destination of the source in the histories of other sources. Transitions measure this relation between consecutive interactions. Windowed co-occurrence also includes destinations separated by intervening events. Both can support a link that the source has never formed. The model stores decayed counts and recent destination lists. It learns only the weights used to combine the resulting estimates. The model has 9–13 learned parameters at context depth $K = 5$, shared across all nodes, and no learned node embeddings. Fixed update rules allow us to compute training features once and reuse them across epochs.

Across 16 TGB and TGB-Seq datasets, the model exceeds EdgeBank and Base3 on all datasets and the heuristic family on 14 under matched event-wise and batch-wise updates. The coverage analysis shows that source-specific transition contexts can be absent while pooled transition evidence remains available. This distinction helps explain prediction beyond repeated source-specific patterns.

Contributions.

- **A statistical model of interaction history.** A log-linear model combines smoothed temporal counts and recency features, with weights shared across nodes and no learned node embeddings (Section 4).
- **Evaluation across two benchmark protocols.** On 16 TGB and TGB-Seq datasets, we compare against recomputed count baselines and reported graph and sequential models (Section 5).
- **Analysis of statistical context.** Controlled comparisons measure the effects of context depth, co-occurrence, and backoff distributions. Context coverage distinguishes source-specific transition evidence from evidence pooled across sources, including where the former is absent (Sections 5.1–6).

2 Related Work

Temporal graph models. JODIE, DyRep, and TGN learn recurrent node states that evolve with interactions (Kumar et al., 2019; Trivedi et al., 2019; Rossi et al., 2020). CAWN and DyGFormer encode temporal walks and neighborhood sequences (Wang et al., 2021; Yu et al., 2023). TPNet combines recent interactions with time-decayed temporal walk counts (Lu et al., 2024). GraphMixer

shows that simpler encoders can also perform well (Cong et al., 2023). These methods learn how to encode interaction history. We use a fixed history representation and learn a shared scoring rule.

Count-based methods. EdgeBank predicts whether an edge has occurred before (Poursafaei et al., 2022). PopTrack uses decayed destination frequencies (Daniluk & Dąbrowski, 2023). Cornell et al. (2025a) combine local and global recency and popularity through a fixed priority order. Base3 interpolates EdgeBank, PopTrack, and a co-occurrence memory (Kondrup, 2025). Our model adds smoothed conditional estimates and learns their contribution to the score.

Sequential recommenders. Session-based methods predict the next item from recent interactions. SKNN uses session co-occurrence (Ludewig & Jannach, 2018). SGNN-HN propagates information through a session graph with learned item embeddings (Pan et al., 2020). CRAFT uses cross-attention between each candidate and the recent interactions of the source (Yi et al., 2026). SGNN-HN and CRAFT provide the strongest reported TGB-Seq scores used in our comparison. Their use of recent sequences motivates our comparison with pooled transition and co-occurrence counts.

Evaluation. Temporal link prediction scores depend on the evaluation protocol. Negative sampling can change method rankings (Poursafaei et al., 2022; Cornell et al., 2025b), sampled metrics can reverse the ordering obtained from full rankings (Krichene & Rendle, 2020), and batch evaluation can withhold recent interactions (Lampert et al., 2026). Analyses of learned models and simple baselines motivate closer control of these factors (Hayes et al., 2025; Bechler-Speicher et al., 2025). We recompute count baselines on the benchmark candidates and measure sensitivity to history-update timing. Context coverage complements the edge-based surprise index of Poursafaei et al. (2022) by measuring whether a transition context has prior observations.

Statistical language modelling. Interpolation and backoff combine estimates from contexts with different amounts of evidence (Katz, 1987; Chen & Goodman, 1999). Unigram interpolation supplies background probabilities from overall word frequencies (MacKay & Peto, 1995). Kneser-Ney smoothing uses the number of distinct preceding contexts to estimate a continuation distribution (Kneser & Ney, 1995). Chen & Goodman (1999) found this approach effective in language modelling. In temporal link prediction, the corresponding counts describe source histories and destination transitions, with exponential decay accounting for their age.

3 Problem setting and evaluation

An interaction stream consists of events (u, v, t), where u is a source, v a destination, and t a timestamp. At a query (u, t), the task is to rank the observed destination above the benchmark negative candidates. Both benchmarks report mean reciprocal rank (MRR) over test queries. We follow their midrank tie rule. The observed destination has rank $1 + n_{>} + \frac{1}{2}n_{=}$, where $n_{>}$ counts negatives with a higher score and $n_{=}$ counts negatives with the same score. Only the rank of the observed destination enters MRR. This treatment matters for count models, which often assign equal scores to unobserved candidates.

TGB-Seq (Yi et al., 2025) provides 100 uniformly sampled negative destinations for each test event. TGB (Huang et al., 2023) uses dataset-specific candidate sets. For example, `tgbl-wiki` uses all possible negative destinations, whereas `tgbl-review` uses 100 sampled negatives per positive edge. Historical-random sampling includes destinations visited by the source during training. Historical negatives test whether a model can distinguish the positive from previously visited destinations. Because candidate distributions differ, we analyse each benchmark separately.

We use the splits, negative candidates, and MRR evaluation supplied by TGB and TGB-Seq. We compare with reported neural benchmark results and recompute the count baselines. Appendix A documents the score sources and evaluation details.

Notation. Throughout, u denotes the source of the current query and z a candidate destination from the destination vocabulary $\mathcal{V}$. The K most recent destinations of u at query time are $v_1, \ldots, v_K$, ordered from most recent. The vocabulary is fixed for each dataset and includes can-

didate destinations first observed after training. t is the query time, and T is the duration of the training split. Probabilities written with a hat, such as $\hat{P}(z \mid u)$, are smoothed estimates (Section 4).

4 A STATISTICAL MODEL FOR TEMPORAL LINK PREDICTION

The model represents history through count tables and recent destination lists. Count estimates describe how often a destination occurs in a particular context. Recency features describe how recently it was visited. A shared log-linear rule combines these two kinds of evidence. The stored counts, timestamps, and destination lists follow fixed update rules, and learning determines the contribution of each feature to the score.

4.1 FROM COMPLETE HISTORIES TO RECURRING CONTEXTS

Complete interaction histories rarely repeat, leaving little evidence for estimating their conditional destination distributions. We use shorter contexts that recur across queries. The marginal $\hat{P}(z)$ estimates destination frequency. The source conditional $\hat{P}(z \mid u)$ estimates the choices of a particular source. We use the last K destinations visited by the source. For each destination v_k, the transition conditional $\hat{P}(z \mid v_k)$ estimates what immediately follows it using transitions observed across all sources. Windowed co-occurrence broadens this context to include pairs separated by intervening interactions. It pools counts over the recent destinations to form the estimate $\hat{P}_{\mathrm{co}}(z \mid v_1, \ldots, v_K)$.

In film recommendation, suppose Alice recently watched films A and B. The marginal estimate describes how popular a film is across all viewers. The source conditional describes how often Alice has watched it. The transition conditionals use what all viewers watched immediately after A or B to estimate her next choice. Co-occurrence also counts films watched within a few choices after A or B. These estimates use different parts of the same history.

4.2 STREAMING COUNT STATE

The count state records destination frequencies, source histories, transitions, and windowed co-occurrences. Each source also retains its recent destination list. Destination counts $c(z; t)$ measure decayed popularity, as in PopTrack (Daniluk & Dąbrowski, 2023). Their total is $C(t) = \sum_{z \in \mathcal{V}} c(z; t)$. Source-destination counts $c(u, z; t)$ record pair history together with the last interaction time. Their keys form the edge memory used by EdgeBank (Poursafaei et al., 2022). Transition counts $c(v, z; t)$ record how often a source moves from v to z in consecutive interactions, pooled across all sources. At query time, each of the last K destinations selects a separate row from this transition table. The model combines these estimates without treating the full sequence as one context.

Windowed co-occurrence counts $c_{\mathrm{co}}(v, z; t)$ record how often z follows v within the last K interactions of the same source. These counts are also pooled across sources. An event updates the count for each destination in the recent window, including repeated occurrences. The table can therefore contain pairs that never occur consecutively, as in session-based recommendation (Ludewig & Jannach, 2018).

Tables store only keys that have occurred. Missing keys have count zero. Each event at time t_i contributes $e^{-(t-t_i)/\tau}$ at query time t, and each count sums these contributions over its matching events. Decay is applied lazily when a cell is accessed. Updating popularity, source history, and consecutive transitions takes $O(1)$ operations per event. Co-occurrence requires up to K count updates. Exponential decay preserves the relation between row entries and their totals, so the normalised estimators remain proper probabilities. The time constant τ is set from timestamps and is not tuned (Appendix G).

4.3 SMOOTHED CONDITIONAL ESTIMATORS

For a context $x \in \{u, v_1, \ldots, v_K\}$, let $n(x; t) = \sum_{z \in \mathcal{V}} c(x, z; t)$ denote its total decayed count. Each context contributes one smoothed estimator. We add pseudocounts from a base distribution B,

using the Dirichlet smoothing form of MacKay & Peto (1995):

$$\hat{P}(z \mid x) = \frac{c(x, z; t) + \alpha\, B(z)}{n(x; t) + \alpha}, \tag{1}$$

$$B(z) \in \left\{ \hat{P}(z) = \frac{c(z; t) + \alpha}{C(t) + \alpha\,|\mathcal{V}|}, \qquad \hat{P}_{\mathrm{c}}(z) = \frac{N_{1+}(\cdot\, z) + \alpha}{\sum_{z'} N_{1+}(\cdot\, z') + \alpha\,|\mathcal{V}|} \right\}, \tag{2}$$

with $\alpha = 1$, $|\mathcal{V}|$ the size of the destination vocabulary, and $N_{1+}(\cdot\, z) = |\{v : c(v, z; t) > 0\}|$ the number of distinct destinations v after which z has been observed. A context with no observations satisfies $\hat{P}(z \mid x) = B(z)$ exactly, so an empty context uses the base distribution.

Equation 2 defines two backoff distributions. The marginal $\hat{P}(z)$ measures how frequently z occurs. The continuation base $\hat{P}_{\mathrm{c}}(z)$ measures how many distinct destinations have preceded z. A destination that occurs often after only one predecessor can therefore have high frequency but a continuation count of one. The continuation count is the number of observed cells in column z of the transition table. These counts do not decay, so the base retains predecessor types from the full observed history.

The continuation counts come from Kneser-Ney smoothing (Kneser & Ney, 1995; Chen & Goodman, 1999). The resulting base assigns more backoff probability to items observed in many contexts. This matters most when the conditioning row has few observations. Both bases use the same Dirichlet smoothing rule. This isolates the effect of the backoff distribution. The marginal also remains a separate feature under either base.

The co-occurrence estimate combines evidence from the whole recent window. We sum the candidate counts over the corresponding rows, normalize by their total count, and smooth toward B. For $L \leq K$ available history positions,

$$\hat{P}_{\mathrm{co}}(z \mid v_1, \ldots, v_L) = \frac{\sum_{k=1}^{L} c_{\mathrm{co}}(v_k, z; t) + \alpha B(z)}{\sum_{k=1}^{L} \sum_{z'} c_{\mathrm{co}}(v_k, z'; t) + \alpha}. \tag{3}$$

Repeated destinations contribute once per history position. With no co-occurrence observations, Equation 3 returns $B(z)$. Appendix E illustrates each estimator on a worked stream.

4.4 Recency features

Recency describes the time of the latest interaction with a candidate. Source-specific recency uses the last visit by u: $r(u, z) = e^{-(t - t_{\mathrm{last}}(u,z))/(\tau_r T)}$, where T is the training duration and τ_r is a fixed fraction. It is zero if u has never visited z. In the film example, this feature measures how recently Alice watched the candidate film.

Global recency measures how recently any source interacted with a candidate. Its value is $e^{-(t - t_{\mathrm{last}}(z))/(\tau_r T)}$, where $t_{\mathrm{last}}(z)$ is the last interaction time of z across all sources. This is the global-recency signal used by Cornell et al. (2025a). It can favor a film watched recently by someone other than the current viewer.

Multi-scale recency measures the same source-destination interaction as $r(u, z)$ at three fixed scales, $\tau_r/10$, τ_r, and $10\tau_r$. The middle scale is $r(u, z)$, so this representation contributes two further features. Multiple scales allow the model to combine fast and slow recency effects. This choice is motivated by retention curves that decay more slowly than a single exponential (Anderson & Schooler, 1991). The learned weights determine the influence of each scale.

4.5 Learned combination

Let $\boldsymbol{\phi}(u, z, t)$ contain the log-probabilities from the count estimates and the untransformed recency values. We score each candidate with a shared log-linear rule:

$$s(z) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(u, z, t) + \beta. \tag{4}$$

Each feature has one learned weight, and β is a learned bias. The weights are unconstrained. A negative weight makes a larger feature value evidence against a candidate. Probability mixtures instead require nonnegative weights.

Table 1: Test MRR (%) on TGB. Our entries are means and sample standard deviations over five runs. EdgeBank is the stronger of its two variants. Best neural gives the strongest reported neural benchmark score (Appendix A). Bold marks the highest MRR per row.

				Ours		Best neural	
Dataset	EdgeBank	Heuristic	Base3	marginal	continuation	MRR	Method
wiki	64.40	82.06	73.26	**82.84** ±0.04	82.72 ±0.10	82.70	TPNet
uci	32.40	**52.75**	35.22	51.57 ±0.32	51.25 ±0.26	24.50	TNCN
enron	15.60	84.66	48.65	86.82 ±0.01	**86.83** ±0.03	37.90	TNCN
subreddit	59.22	74.19	74.25	**74.90** ±0.06	74.79 ±0.04	69.60	TNCN
lastfm	2.63	16.58	11.37	32.84 ±0.18	**32.87** ±0.12	15.60	TNCN
review	2.69	33.93	12.87	36.66 ±0.74	45.76 ±0.39	**52.10**	GraphMixer
coin	58.31	81.10	77.71	82.09 ±0.06	81.65 ±0.02	**88.47**	CRAFT-R
comment	15.03	72.38	45.41	47.94 ±1.27	54.76 ±0.87	**91.72**	CRAFT
flight	38.71	88.92	79.34	89.93 ±0.06	89.99 ±0.04	**91.39**	CRAFT-R

If a source has fewer than K past interactions, transition features for unavailable history positions are zero for every candidate. These inactive features do not affect the ranking. An available destination with no observed successors instead uses the backoff distribution in Equation 1. Figure 1 shows how individual features contribute to a candidate score.

The evaluated configurations have 9–13 learned parameters at $K = 5$, including the bias. Changing the backoff distribution adds no parameters. There are no node or item embeddings, message passing, or learned per-node state. A source enters through its recent destinations and count rows. The same scoring rule applies to nodes first observed at test time. The statistical state grows with the number of observed destinations and pairs, so the parameter count alone does not describe the memory cost.

4.6 Fitting

Features are extracted in a single chronological pass, before adding each event to history. The resulting vectors are reused across epochs and shuffled into minibatches to fit only the scoring weights. Training minimises a sampled softmax loss with one positive and one uniformly sampled negative destination per query. The number of negatives per positive follows TGN (Rossi et al., 2020, Appendix A.4). Test candidates are provided by the benchmark.

5 Results and discussion

Setup. We evaluate on 9 TGB and 7 TGB-Seq datasets. Main comparisons use five runs. Feature, depth, and smoothing analyses use three runs. Timing comparisons reuse the five main fits. Tables report means and sample standard deviations. Hyperparameters are fixed as specified in Section 4 and Appendix G. Tables 1 and 2 report TGB and TGB-Seq separately.

We fix one feature configuration per dataset and backoff distribution, as listed in Appendix G. These feature sets were chosen on validation data before the reported fits. Mean validation MRR over five runs chooses between the two backoff configurations. We also choose the heuristic rule on validation MRR. All performance rankings use test MRR. For controlled comparisons, the *reference configuration* contains popularity, source history, the K transition estimates, and source-specific recency at scale τ_r. We add co-occurrence, global recency, or two further source-recency scales separately and together, under each backoff. These comparisons use the same three seeds for both configurations.

Following the streaming setting of TGB, observed test interactions enter history for later predictions while learned weights remain fixed (Huang et al., 2023, Appendix C). Our history updates use fixed count operations, making event-wise evaluation practical. We update statistics after each scored event, retaining file order for tied timestamps. Our model, EdgeBank, Base3, and the heuristic family use the same candidates and event-wise updates. Appendix I examines batch-wise and strict-timestamp updates.

Table 2: Test MRR (%) on TGB-Seq. Columns follow Table 1. Count baselines use the TGB-Seq candidate sets.

				Ours		Best neural	
Dataset	EdgeBank	Heuristic	Base3	marginal	continuation	MRR	Method
GoogleLocal	1.96	18.39	3.97	31.27 ±0.02	31.42 ±0.03	**62.88**	SGNN-HN
Yelp	9.77	30.81	11.99	53.21 ±0.06	52.16 ±0.09	**72.69**	CRAFT
Taobao	20.28	42.99	20.17	59.58 ±0.03	58.90 ±0.06	**70.68**	CRAFT
ML-20M	1.94	17.71	6.64	**39.24** ±0.15	39.00 ±0.10	35.91	CRAFT
Flickr	1.96	38.38	34.81	**62.68** ±0.07	62.66 ±0.08	62.34	CRAFT
YouTube	1.96	56.64	38.09	**65.27** ±0.06	65.12 ±0.04	59.64	SGNN-HN
WikiLink	1.96	47.70	26.61	67.95 ±0.05	67.97 ±0.04	**75.48**	CRAFT

The model achieves the highest MRR among all compared methods on 7 of 16 datasets. Each of these leads holds in all five runs. CRAFT and CRAFT-R together lead on 6 datasets, the next highest count. On TGB, the lead over TPNet on tgbl-wiki is 0.14 points. The largest gains over reported TGB references occur on tgbl-uci, tgbl-enron, and tgbl-lastfm. Recomputed heuristics are also strong on these datasets, and exceed our model on tgbl-uci. On TGB-Seq, the model leads on ML-20M, Flickr, and YouTube. These datasets have no previously observed source-specific transition contexts at test time (Section 6).

The model remains below the reported references on 4 TGB and 4 TGB-Seq datasets. The largest deficits within each benchmark occur on tgbl-comment and GoogleLocal. Against recomputed count methods, it exceeds EdgeBank and Base3 on 16 of 16 datasets and the heuristic rule on 14 of 16. These gains persist when all counting methods use batch-200 updates. Under this schedule, our model still achieves the highest MRR in the comparison on 4 of 16 datasets (Appendix I).

5.1 The effect of the backoff distribution

We compare marginal and continuation backoff with the reference configuration fixed at $K = 5$. Both models are fitted with the same three seeds and training settings.

The three largest backoff effects occur on datasets with low source-specific transition coverage (Section 6). Continuation backoff improves MRR by 9.13 points on tgbl-review and 3.83 points on tgbl-comment (Figure 2). Marginal backoff is stronger on Yelp by 5.01 points. On the remaining datasets, the absolute difference is at most 1.39 points.

Continuation backoff favors destinations reached from many distinct predecessors. Its gains show the value of this distinction from raw frequency. Sparse history alone therefore does not determine the better background distribution.

5.2 The contribution of co-occurrence and recency

Adding co-occurrence to the reference configuration gives a larger median gain than adding global recency or further recency scales, under either backoff distribution. With marginal backoff, the median gain is 0.82 MRR points (Appendix F). Relative to the reference configuration with the same backoff distribution, the main model gains 1.44 points at the median. Its largest gain is 13.01 points on Yelp.

The co-occurrence gains suggest that useful associations between destinations extend beyond immediate succession. Pooling these observations across the recent window supplies evidence that consecutive transition counts omit.

5.3 The learned scoring rule

We examine mean coefficients with the reference configuration and marginal backoff on every dataset. Features differ in scale, so we interpret signs and relative weights within the transition features. Appendix C reports coefficients for both backoffs and all feature additions.

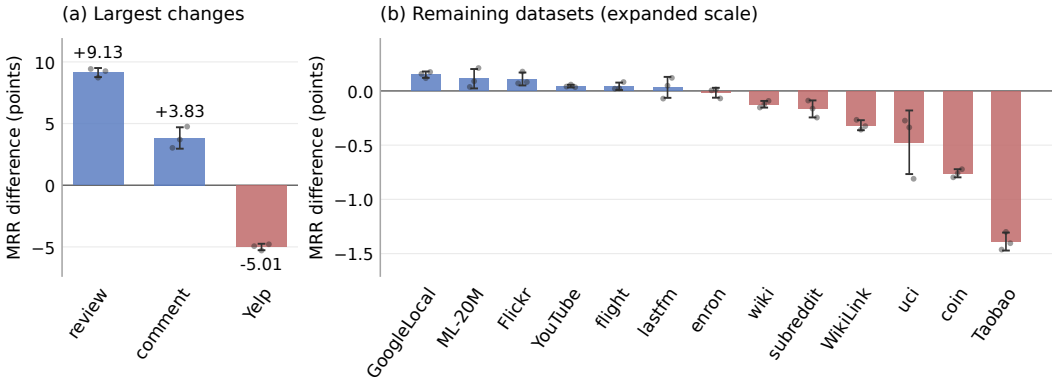

Figure 2: Effect of changing marginal to continuation backoff with the reference configuration fixed at $K = 5$. Bars show mean test MRR differences across three paired runs. Dots show individual runs, and whiskers show one sample standard deviation. Positive values favor continuation. Panel (a) shows the three largest absolute mean changes. Panel (b) uses an expanded linear scale for the remaining datasets.

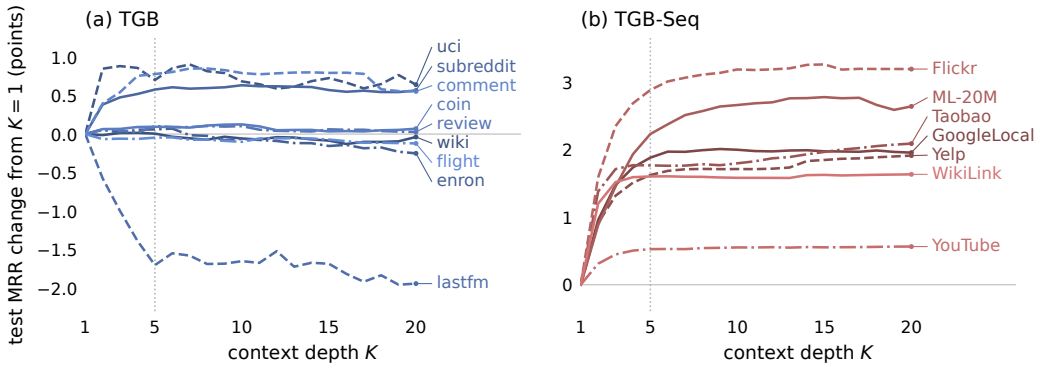

Figure 3: Context depth with marginal backoff and the reference configuration. Lines show mean test MRR changes from $K = 1$ across three runs. The dotted line marks $K = 5$. Panels use different vertical scales.

The source-history coefficient is positive on all 9 TGB datasets, and recency is positive on 8. On TGB-Seq, both coefficients are negative on 5 of 7 datasets. Source history can thus play two roles: its counts can discourage a return to a visited destination, while its recent destinations still provide contexts for predicting a new link through pooled transitions. The most recent destination has the largest transition coefficient on 16 of 16 datasets.

5.4 The value of longer contexts

We fit the reference configuration with marginal backoff separately at each depth from $K = 1$ to $K = 20$. Increasing K from one to 20 improves mean test MRR on all 7 TGB-Seq datasets, with gains in every seed (Figure 3). Most of the improvement occurs within the first 5 destinations. On these datasets, extending the context from 5 to 20 adds at most 0.41 MRR points. On TGB, the same increase from one to 20 improves mean MRR on 5 of 9 datasets. The smaller gains beyond $K = 5$ and the mixed TGB results favor a short recent history as a practical default. Appendix D gives the sweep protocol.

6 When is context available?

Coverage measures whether a context has prior observations when a query is scored. For a source u with previous destination v_1, we distinguish two kinds of evidence. A pooled transition is available if any source has previously moved from v_1 to another destination. A source-specific transition is available if u has previously done so. Histories follow the event ordering in Section 5.

Let $n_u(v_1; t)$ be the decayed count of transitions from v_1 contributed by source u, so $n(v_1; t) = \sum_{u'} n_{u'}(v_1; t)$. Here $v_1 = v_1(u, t)$ depends on the source and query time. Over the test queries Q, *pooled coverage* is the fraction $|\{(u, t) \in Q : n(v_1; t) > 0\}|/|Q|$. *Pair coverage* is the fraction $|\{(u, t) \in Q : n_u(v_1; t) > 0\}|/|Q|$. The model uses pooled transitions, with exact backoff for empty rows (Equation 1).

Pooled transition evidence is available on at least 96.1% of test queries on every dataset. By contrast, 5 TGB-Seq datasets have zero pair coverage. On `ML-20M`, pooled coverage is complete, so transitions observed for other sources remain available.

Figure 4 relates context availability to performance. Among the datasets with zero pair coverage, the model exceeds the reported references on `ML-20M`, `Flickr`, and `YouTube`, but remains below them on `GoogleLocal` and `WikiLink`. At the other end of the coverage range, `tgbl-coin` and `tgbl-flight` remain below their references despite pair coverage of at least 87.1%. Context availability alone does not determine performance.

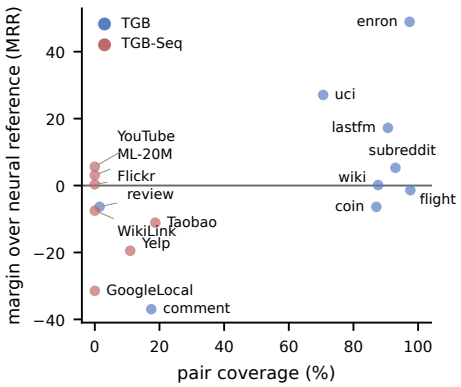

Figure 4: Source-specific transition coverage against the MRR margin over the neural reference, one point per dataset.

Pair coverage concerns a particular transition context, not the entire source history. A learned node memory can still encode earlier interactions when pair coverage is zero. Coverage also differs from the surprise index, which measures whether a test edge occurred during training (Poursafaei et al., 2022). Suppose a source previously visited A followed by B and has now returned to A. Its next destination C can be new to the source even though the transition context is familiar. Pair coverage also includes observations after training.

7 Conclusion

Our work shows that statistical language modelling can predict temporal links beyond repeated edges. The proposed model draws on transition and co-occurrence patterns pooled across sources. These patterns support new links even when the current source has not encountered the same transition before. In our main evaluation, the model achieves the highest MRR among the compared methods on 7 out of 16 datasets from TGB and TGB-Seq. It also outperforms EdgeBank and Base3 on all 16 datasets and the heuristic family on 14. These results show that fixed statistical summaries can provide a competitive alternative to learned node states.

Limitations and future work. The timing analysis keeps the configuration and weights fixed (Appendix I). Future work could test whether choosing marginal or continuation backoff separately for each query improves prediction.

With only 9–13 learned parameters, our model provides a simple and competitive baseline against which future neural temporal link predictors should be evaluated.

Reproducibility statement

Tables, figures, and numerical results in the text are generated from the per-run result files and recorded reference scores. Our count baselines use the same candidate sets as the model. Code is available at https://doi.org/10.5281/zenodo.22914326. We use the public TGB and TGB-Seq distributions with their supplied splits and negative candidates.

AI use statement

The authors take responsibility for the accuracy and originality of this work, including its methods, results, text, and accompanying artifacts. We used generative AI tools to assist with method and code development, experimental design, figure preparation, and manuscript revision. Reported numerical results are generated from experiment outputs. Verification includes automated tests, checks against recorded results, and manual code inspection.

References

John R. Anderson and Lael J. Schooler. Reflections of the environment in memory. *Psychological Science*, 2(6):396–408, 1991. URL https://doi.org/10.1111/j.1467-9280.1991.tb00174.x.

Maya Bechler-Speicher, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M. Bronstein, Mathias Niepert, Bryan Perozzi, Mikhail Galkin, and Christopher Morris. Position: Graph learning will lose relevance due to poor benchmarks. In *Proceedings of the 42nd International Conference on Machine Learning, Position Paper Track*, 2025. URL https://proceedings.mlr.press/v267/bechler-speicher25a.html.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999. URL https://doi.org/10.1006/csla.1999.0128.

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 *International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=ayPPc0SyLv1.

Filip Cornell, Oleg Smirnov, Gabriela Zarzar Gandler, and Lele Cao. On the power of heuristics in temporal graphs. In *Proceedings on "I Can't Believe It's Not Better: Challenges in Applied Deep Learning" at ICLR 2025 Workshops*, volume 296 of *Proceedings of Machine Learning Research*, pp. 37–46, 2025a. URL https://proceedings.mlr.press/v296/cornell25a.html. arXiv:2502.04910.

Filip Cornell, Oleg Smirnov, Gabriela Zarzar Gandler, and Lele Cao. Are we really measuring progress? transferring insights from evaluating recommender systems to temporal link prediction. In *Proceedings of the Temporal Graph Learning Workshop at SIGKDD*, 2025b. URL https://openreview.net/forum?id=S6BfBrrD9L. arXiv:2506.12588.

Michał Daniluk and Jacek Dąbrowski. Temporal graph models fail to capture global temporal dynamics. In *Temporal Graph Learning Workshop at NeurIPS*, 2023. URL https://arxiv.org/abs/2309.15730. arXiv:2309.15730.

Abigail J. Hayes, Tobias Schumacher, and Markus Strohmaier. What do temporal graph learning models learn? *arXiv preprint arXiv:2510.09416*, 2025. URL https://arxiv.org/abs/2510.09416.

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 *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2307.01026. arXiv:2307.01026.

Shenyang Huang, Ali Parviz, Emma Kondrup, Zachary Yang, Zifeng Ding, Michael Bronstein, Reihaneh Rabbany, and Guillaume Rabusseau. Are large language models good temporal graph learners? *arXiv preprint arXiv:2506.05393*, 2025. URL https://arxiv.org/abs/2506.05393.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3): 400–401, 1987. URL https://doi.org/10.1109/TASSP.1987.1165125.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pp. 181–184, 1995. URL https://doi.org/10.1109/ICASSP.1995.479394.

Emma Kondrup. Base3: A simple interpolation-based ensemble method for robust dynamic link prediction. In *Proceedings of the Temporal Graph Learning Workshop at KDD*, 2025. URL https://arxiv.org/abs/2506.12764. arXiv:2506.12764.

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1748–1757, 2020. URL https://research.google/pubs/on-sampled-metrics-for-item-recommendation/.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1269–1278, 2019. doi: 10.1145/3292500.3330895. URL https://doi.org/10.1145/3292500.3330895.

Moritz Lampert, Christopher Blöcker, and Ingo Scholtes. From link prediction to forecasting: Addressing challenges in batch-based temporal graph learning. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2406.04897. arXiv:2406.04897.

Xiaodong Lu, Lu Yi, and Zhewei Wei. Improving temporal link prediction via temporal walk matrix projection. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. URL https://arxiv.org/abs/2410.04013. arXiv:2410.04013.

Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28:331–390, 2018. URL https://arxiv.org/abs/1803.09587.

David J. C. MacKay and Linda C. Bauman Peto. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):289–308, 1995. URL https://www.cambridge.org/core/journals/natural-language-engineering/article/abs/hierarchical-dirichlet-language-model/07CB63E866B2386854A1CA5BAA30055D.

Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. Star graph neural networks for session-based recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 1195–1204, 2020. doi: 10.1145/3340531.3412014. URL https://doi.org/10.1145/3340531.3412014.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022) Datasets and Benchmarks Track*, volume 35, pp. 32928–32941, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/d49042a5d49818711c401d34172f9900-Abstract-Datasets_and_Benchmarks.html.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020. URL https://arxiv.org/abs/2006.10637. arXiv:2006.10637.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyePrhR5KX.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KYPz4YsCPj. arXiv:2101.05974.

Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Zixuan Yue, and Zengfeng Huang. TGB-Seq benchmark: Challenging temporal GNNs with complex sequential dynamics. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=8e2LirwiJT. arXiv:2502.02975.

Lu Yi, Runlin Lei, Fengran Mo, Yanping Zheng, Zhewei Wei, and Yuhang Ye. Future link prediction without memory or aggregation. *Advances in Neural Information Processing Systems*, 38:2347–2375, 2026. doi: 10.52202/085713-0084. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/036912a83bdbb1fd792baf6532f102d8-Paper-Conference.pdf.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. URL https://arxiv.org/abs/2303.13047. arXiv:2303.13047.

Xiaohui Zhang, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Efficient neural common neighbor for temporal graph link prediction. In *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, 2025. URL https://arxiv.org/abs/2406.07926. arXiv:2406.07926.

A Datasets and reference scores

Scope. We evaluate the 9 TGB link-prediction datasets and 7 TGB-Seq datasets, each under the shipped candidate sets of its benchmark and evaluator.

Sources of the neural reference scores. The leaderboards are https://tgb.complexdatalab.com/docs/leader_linkprop/ and https://tgb-seq.github.io/leaderboard/, last accessed September 18, 2026. The reference columns contain neural methods. Counting baselines appear in their separate columns under matched evaluation schedules. For tgbl-wiki and tgbl-review, we use the leaderboard scores for TPNet and GraphMixer, respectively. For tgbl-coin, tgbl-comment, and tgbl-flight it is CRAFT, from Tables 2 and 3 of Yi et al. (2026), evaluated on TGB's shipped negatives with baseline rows that match the leaderboard, and stronger than any leaderboard entry. For tgbl-uci, tgbl-enron, tgbl-lastfm, and tgbl-subreddit it is TNCN (Zhang et al., 2025) as reported by Huang et al. (2025).

Reproduction of EdgeBank. We evaluate unlimited memory and the TGB time window, whose duration is fifteen percent of the training time span. The main tables use event-wise updates. With batch-200 updates, both variants match the reported scores at their displayed precision on all five current TGB leaderboard datasets. On tgbl-wiki, unlimited and windowed memory give 49.47 and 57.10, compared with the reported 49.50 and 57.10.

Reproduction of the heuristic family. We implement the four heuristics of Cornell et al. (2025a) using the same candidates, tie rule, and event ordering as our model. This extends their evaluation to 10 additional datasets. Our implementation reproduces their combined score within one point on five of the six TGB datasets they report. On tgbl-review, their evaluation uses a dataset-specific inverse-recency variant. The combination stated in their method section obtains 17.69 in our implementation, compared with their reported 52.20. The result files include scores for every individual heuristic. Appendix I compares the same heuristic rule under event-wise and batch-wise updates.

Reproduction of Base3. We implement Base3 from the released code and evaluate both update schedules. The main tables use event-wise updates. With batch-200 updates, our results are within

half a point of the reported scores on `tgbl-comment`, `tgbl-flight`, and `tgbl-coin`. They are lower on `tgbl-review` and `tgbl-wiki`. The released code refers to negative-candidate files that are absent from the repository and specifies paths for a single dataset. We therefore cannot reconstruct the exact candidates used for those reported results. Our comparisons use the candidates supplied by TGB.

TNCN reference scores. For `tgbl-uci`, `tgbl-enron`, `tgbl-lastfm`, and `tgbl-subreddit`, we use the TNCN results reported by Huang et al. (2025). Their evaluation measures MRR using negatives generated according to the TGB procedure. Our evaluation uses the candidate files distributed with TGB. The comparisons therefore share a sampling procedure, but may differ in the particular candidates and the timing of history updates.

TGB-Seq references. The neural reference scores come from Table 2 of Yi et al. (2026), which reports CRAFT and SGNN-HN on every TGB-Seq dataset.

B THE COMPARED METHODS

EdgeBank, the heuristic family, and Base3 are recomputed in our experiments. The other methods are compared through their reported results. Appendix A gives the source of each score. Our model stores history in count tables with fixed update rules and learns the weights of smoothed estimates and recency features. The methods below differ in their history representations, update rules, and scoring functions.

JODIE (Kumar et al., 2019) assigns each user and item a static and a dynamic embedding. Two recurrent networks update the dynamic embeddings at each interaction using the counterpart, interaction features, and elapsed time. A learned projection evolves an embedding between interactions, and a linear layer predicts the next item embedding.

DyRep (Trivedi et al., 2019) models long-lived associations and transient communications with temporal point processes. A recurrent architecture updates node embeddings at each event. A learned intensity over the endpoint embeddings determines the likelihood of an interaction. Embeddings continue to update during evaluation.

TGN (Rossi et al., 2020) maintains a memory vector for each node. Events generate messages that are aggregated and passed to a learned memory updater. A temporal embedding module combines this memory with neighborhood information. Memory continues to update during evaluation.

SGNN-HN (Pan et al., 2020) constructs a graph from the items in a session and connects them through a central node. A gated GNN propagates information through this graph. A highway gate combines the initial and updated item embeddings. An attention-based session representation then scores candidate items. TGB-Seq retains only nodes seen during training for all compared methods (Yi et al., 2025).

CAWN (Wang et al., 2021) samples walks backward in time from both endpoints of a query. It replaces node identities with occurrence counts relative to the sampled walks. An RNN encodes the anonymised walks and their time gaps, and an MLP scores the pair from these encodings.

EdgeBank (Poursafaei et al., 2022) stores observed source-destination pairs. It predicts an edge as positive if that pair is in memory. The unlimited variant retains every observed pair. The time-window variant retains pairs from a recent interval. We use the window rule in the TGB reference implementation for this variant.

GraphMixer (Cong et al., 2023) uses an MLP-Mixer to encode recent links with fixed cosine time features. A separate node encoder averages neighbor features over a time window. An MLP combines the two representations to score a pair. The method learns a representation of recent interactions without recurrent networks or self-attention.

DyGFormer (Yu et al., 2023) encodes the first-hop interaction sequences of both query nodes. Features include neighbor attributes, time, and the frequency with which neighbors occur in both sequences. The sequences are divided into patches and processed by a transformer. An MLP scores the pair from the resulting representations.

PopTrack (Daniluk & Dąbrowski, 2023) maintains decayed destination counts. It scores a batch from the current counts, then updates and decays them. The decay factor is selected on validation.

TPNet (Lu et al., 2024) maintains random projections of time-decayed temporal walk counts. Learned MLPs combine the recovered walk features with recent-interaction encodings to score each pair.

TNCN (Zhang et al., 2025) combines node memory with neural common-neighbor features. Events update the memories, and a graph transformer computes embeddings from temporal neighborhoods. The pair score combines endpoint embeddings with information from temporal common neighbors.

The heuristic family of Cornell et al. (2025a) uses local and global recency and popularity. The first statistic in a fixed priority order ranks candidates. Subsequent statistics break remaining ties.

Base3 (Kondrup, 2025) combines EdgeBank membership, PopTrack popularity, and a windowed co-occurrence memory. The co-occurrence component uses decayed neighbor popularity and saturating counts. Binary confidence indicators determine the interpolation weights through a fixed lookup.

CRAFT (Yi et al., 2026) learns node embeddings and uses cross-attention from each candidate to the recent neighbors of the source. It includes positional information and the time since the candidate was last active. An MLP produces the final score. CRAFT-R is its variant for datasets dominated by repeated edges and provides the reference scores on `tgbl-coin` and `tgbl-flight`.

C THE FITTED WEIGHTS

Figure 5 reports the mean coefficients of the reference configuration under both backoff distributions. Figure 6 reports the weights of co-occurrence, global recency, and further recency scales when each is added separately. All means use three runs. A larger weight does not always mean a stronger effect on the score. Recency and log-probability features have different numerical ranges. We therefore compare weight sizes only for features of the same kind, such as transition estimates at different history positions.

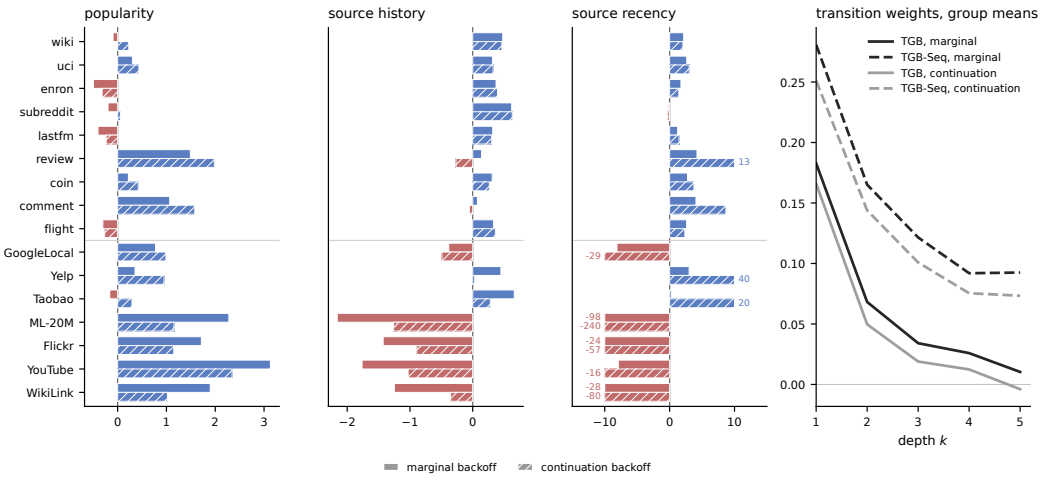

Figure 5: Mean fitted weights of the reference configuration over three runs. Plain bars use marginal backoff and hatched bars use continuation backoff. TGB datasets appear above the divider. Blue denotes positive weights and red negative weights. Recency weights beyond ± 10 are printed beside the clipped bars. Right: mean transition coefficients at each history position, grouped by benchmark and backoff distribution.

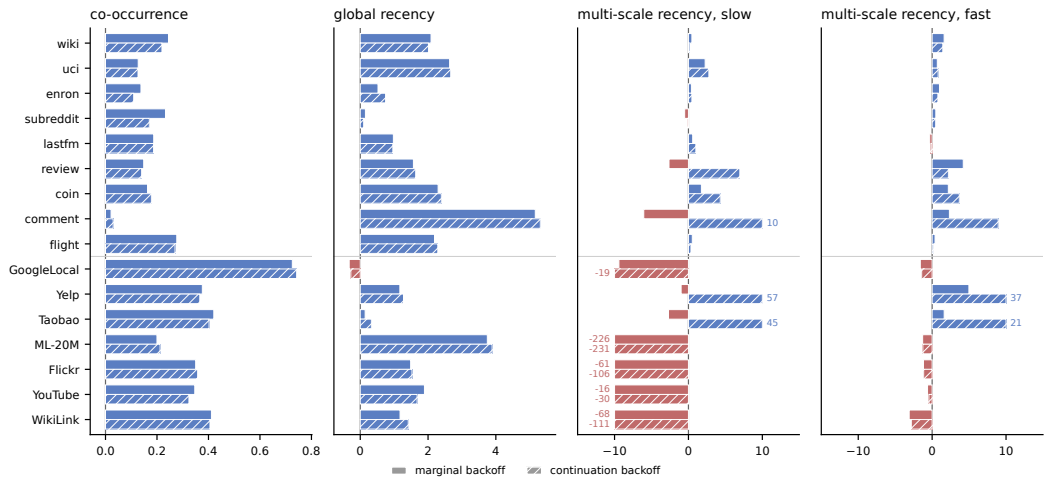

Figure 6: Mean weights of co-occurrence and recency features when added to the reference configuration, over three runs. Backoff distributions follow Figure 5. The two multi-scale panels show the additional slow ($10\tau_r$) and fast ($\tau_r/10$) recency features. Weights beyond ± 10 are printed beside the clipped bars.

D Sensitivity to the context depth

The sweep uses the reference configuration with marginal backoff and three runs on every dataset. It varies K from one to 20 while holding the smoothing, decay, and recency settings fixed. The context depth is fixed at $K = 5$ in the main experiments. A model at depth k uses popularity, source history, source recency, and the first k transition features. We extract features once at depth 20, fit a separate scoring rule for each prefix, and evaluate all depths in one pass. At $K = 5$, we reuse the corresponding reference fit. Each fit uses every training query, at most 100 epochs, and validation patience 10. The sweep measures sensitivity within this configuration and does not change the configurations reported in the main tables. The largest depth equals the smallest sampled-neighbor budget in the TGB-Seq baseline grids, which range from 20 to 60 neighbors (Yi et al., 2025).

The TGB-Seq gains from $K = 1$ to $K = 20$ range from 0.57 to 3.19 MRR points. The largest gain occurs on `Flickr`, where source-specific transition coverage is zero (Section 6). The model can still draw on earlier destinations because its transition estimates pool observations across sources. On TGB, mean MRR increases on `tgbl-uci`, `tgbl-subreddit`, `tgbl-review`, `tgbl-coin`, and `tgbl-comment`, while it decreases on the remaining datasets. The largest decline is 1.94 points on `tgbl-lastfm`.

E A worked example of the estimators

The six-event stream in Figure 1 gives a concrete example of each estimator. We use $\alpha = 1$, $K = 2$, and undecayed counts. The vocabulary is $\{a, b, c, d\}$, so $C(t) = 6$ and $|\mathcal{V}| = 4$. Candidate b occurs twice, giving $\hat{P}(b) = (2 + 1)/(6 + 4) = 0.30$. Both events belong to u, which has five events in total. The source conditional is therefore $\hat{P}(b \mid u) = (2 + 0.30)/(5 + 1) = 0.38$.

The most recent destination is $v_1 = c$. There is one observed transition from c, and it leads to b. Thus $\hat{P}(b \mid c) = (1 + 0.30)/(1 + 1) = 0.65$. For an unobserved successor such as a, smoothing gives $\hat{P}(a \mid c) = (0 + 0.20)/(1 + 1) = 0.10$. The transition row of d is empty, so $\hat{P}(z \mid d) = \hat{P}(z)$ for every candidate.

The continuation base changes this fallback distribution. Candidate b has followed 2 distinct predecessors, whereas c has followed 1. Neither a nor d has an observed predecessor. The total continuation count is 3, and $\hat{P}_c(b) = (2 + 1)/(3 + 4) = 0.43$. Although b and c occur equally often,

the continuation base favors b because it follows more distinct predecessors. Under continuation backoff, the empty row of d returns $\hat{P}_{\mathrm{c}}$.

Table 3 gives all candidate estimates. The two recent destinations favor different candidates. The learned score combines these preferences. Smoothing also assigns positive probability to d, which u has never visited.

Table 3: Smoothed estimates on the worked stream of Figure 1, computed by the model implementation ($\alpha = 1$, counts undecayed). Conditional rows use marginal backoff. Bold marks the candidates each estimator favours.

Estimator	a	b	c	d
$\hat{P}(z)$	0.20	**0.30**	**0.30**	0.20
$\hat{P}_{\mathrm{c}}(z)$	0.14	**0.43**	0.29	0.14
$\hat{P}(z \mid u)$	0.20	**0.38**	**0.38**	0.03
$\hat{P}(z \mid v_1{=}c)$	0.10	**0.65**	0.15	0.10
$\hat{P}(z \mid v_2{=}b)$	0.07	0.10	**0.77**	0.07

Figure 7 illustrates windowed co-occurrence (Equation 3) and the recency features. The co-occurrence table records (a, c) from nonconsecutive interactions at t_1 and t_3. Global recency distinguishes candidate d, which another source visited but u did not. Multi-scale recency applies three fixed decay scales to the time since the source last visited each candidate. The features describe complementary aspects of the stream: associations within a window, activity across sources, and time since the source last visited a candidate.

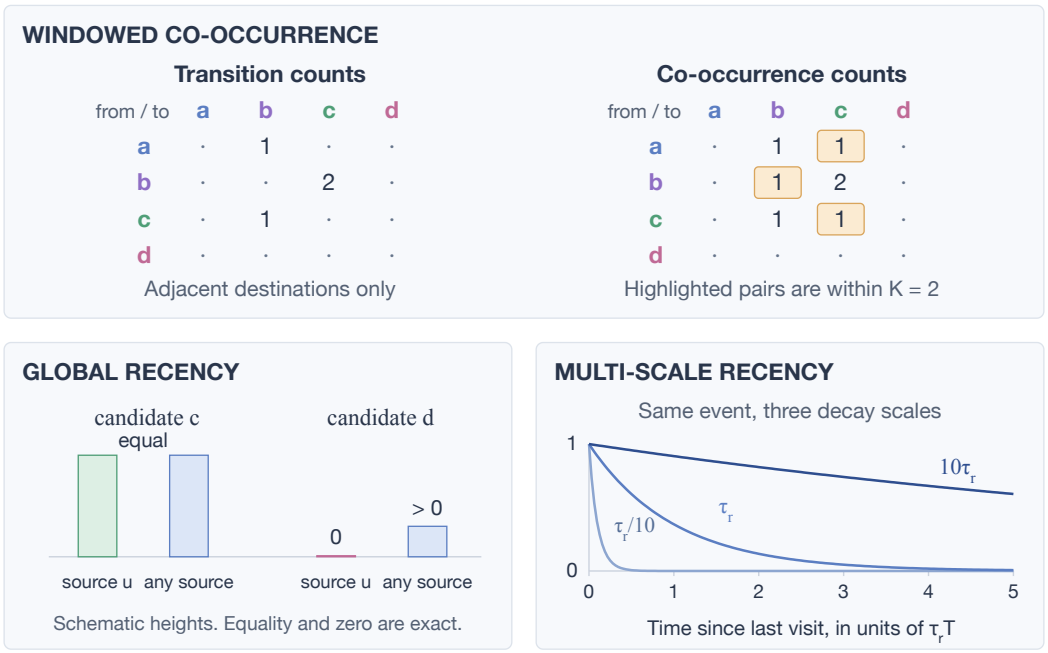

Figure 7: Windowed co-occurrence and recency features on the worked stream of Figure 1. Top: the transition and co-occurrence tables the six events produce. Each event with prior source history updates one transition cell and up to K co-occurrence cells, one for each previous destination in the window. Dots mark unobserved pairs. The highlighted cells, (a, c) from t_1 and t_3, (b, b) from t_2 and t_4, and (c, c) from t_3 and t_5, contain pairs that occur within the window but never consecutively. Bottom left: the two recency features for candidates c and d. The equality for c is exact because its most recent interaction is with u. The source recency of d is zero because u has never visited it. Other bar heights are schematic. Bottom right: the three fixed retention curves of multi-scale recency, independent of the stream. Their learned weights determine how the time since the source last visited a candidate affects its score.

F Controlled feature comparisons

We add co-occurrence, global recency, and further source-recency scales to the reference configuration, first separately and then together. We repeat these comparisons under both backoff distributions, using three runs per configuration. Figure 8 compares each addition with the reference on the same backoff distribution. Co-occurrence gives the largest median gain among the individual additions. With marginal backoff, it improves MRR on 14 of 16 datasets. Combining all features has a larger median gain, but improves fewer datasets. Thus, adding more features does not consistently improve performance.

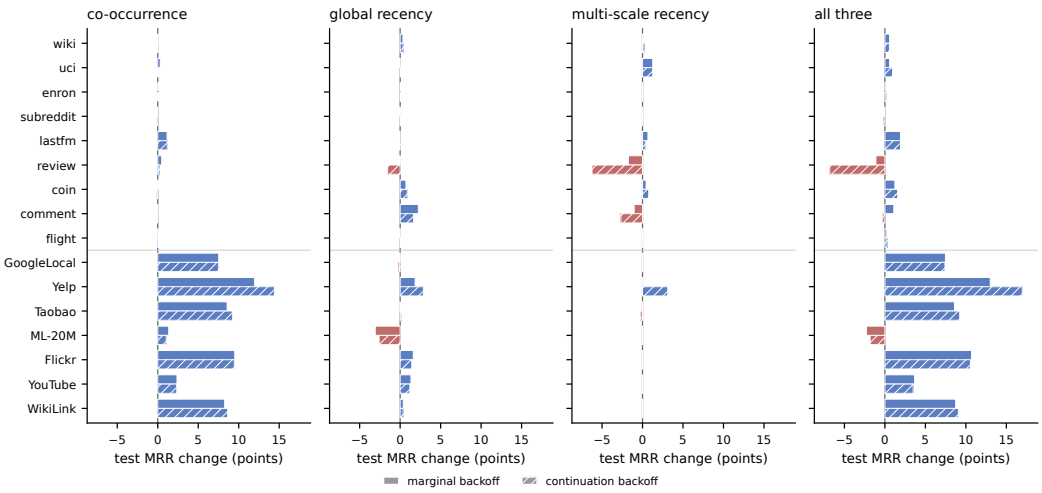

Figure 8: Change in mean test MRR from adding co-occurrence, global recency, further source-recency scales, or all three to the reference configuration. Each addition is compared under the same backoff distribution. Each comparison uses three runs. Plain bars use marginal backoff and hatched bars use continuation backoff. TGB datasets appear above the divider.

G HYPERPARAMETERS AND THEIR PROVENANCE

The EdgeBank paper sets the recent-history window to the validation duration (Poursafaei et al., 2022, Section 4). We use the same duration for the count-decay constant τ. Recency uses the training duration T. Each duration is the last timestamp minus the first timestamp in that split. Both constants are fixed during fitting and evaluation. The main comparisons use $\alpha = 1$. Appendix H examines sensitivity to this choice. The context depth $K = 5$ and recency constant $\tau_r = 0.01$ are also fixed. Appendix D examines context depth, and Appendix F evaluates additional recency scales.

Table 4: Feature sets used in the main comparisons. Entries name additions to the reference configuration. These choices remain fixed across five runs.

Dataset	Marginal backoff	Continuation backoff
`wiki`	all three	all three
`uci`	all three	all three
`enron`	all three	all three
`subreddit`	reference	global recency
`lastfm`	all three	all three
`review`	co-occurrence	co-occurrence
`coin`	all three	all three
`comment`	multi-scale recency	global recency
`flight`	all three	all three
`GoogleLocal`	co-occurrence	co-occurrence
`Yelp`	all three	all three
`Taobao`	co-occurrence	co-occurrence
`ML-20M`	co-occurrence	co-occurrence
`Flickr`	all three	all three
`YouTube`	all three	all three
`WikiLink`	all three	all three

The reference configuration has $K + 4$ learned parameters, including the bias. Co-occurrence and global recency each add one parameter. Multi-scale recency adds two, and the configuration with all features has $K + 8$.

The two decay constants apply to different quantities. The count tables use τ, measured in the timestamp units of the data. The recency feature uses $\tau_r T$, a fixed fraction of the training duration.

The scoring rule is fitted with Adam at learning rate 10^{-2} for at most 100 epochs, using batches of 256 queries and one uniform negative per query. Gradients are clipped to a norm of one. We use no weight decay. We measure validation MRR after each epoch and stop after ten consecutive epochs without improvement. Test evaluation uses the weights from the epoch with the highest validation MRR. Validation candidates remain fixed across epochs. Every training event supplies a training query. Its features are computed before the event updates the count tables. Every main-model fit stops by the patience criterion before reaching the epoch limit.

H SENSITIVITY TO SMOOTHING

We vary α over $\{10^{-1}, 1, 10\}$ with marginal backoff, the reference configuration, and $K = 5$. The same value is used in the marginal and conditional estimators. Each setting uses three runs on all 16 datasets. The $\alpha = 1$ results reuse the marginal reference fits from the feature comparisons. This comparison examines sensitivity within a fixed configuration. It does not select α for the main results. Across the three smoothing constants, the median range of mean test MRR is 0.75 points. The largest range is 3.03 points on `tgbl-review` (Figure 9).

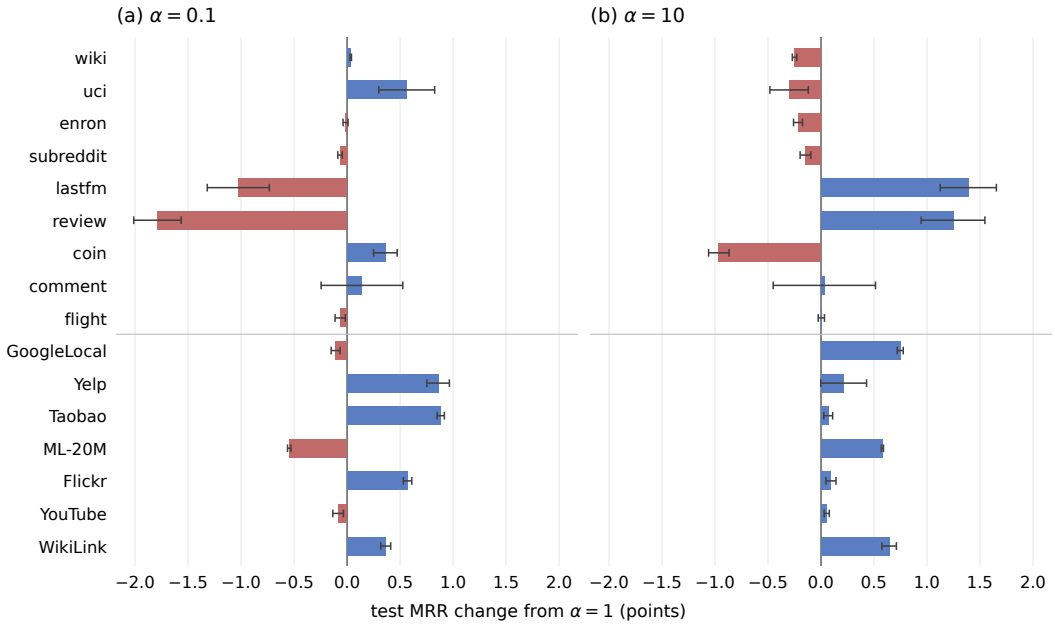

Figure 9: Change in test MRR from $\alpha = 1$ under smaller and larger smoothing constants. Bars show mean changes over three seeds. Error bars show one sample standard deviation of the changes, with runs paired by seed. Blue denotes increases and red denotes decreases. TGB datasets appear above the divider. All settings use marginal backoff, the reference configuration, and $K = 5$.

I SENSITIVITY TO EVALUATION TIMING

We vary evaluation history while fixing the model configuration and weights. All conditions use the same five fitted models and test candidates for each dataset. EdgeBank, Base3, and the heuristic family are also evaluated under event-wise and batch-wise updates. The heuristic rule remains fixed across schedules.

Event-wise evaluation updates history after each scored event, retaining file order for tied timestamps. *Batch-wise* evaluation scores 200 consecutive test events before adding any of them to history. *Strict-timestamp* evaluation scores all events at time t using only events with earlier timestamps, including training and validation events. After scoring a group, we update the tables in file order. The transition rule is unchanged, but earlier rows at the query timestamp are excluded from its history.

The released EdgeBank and TGB implementations update memory after batches of 200 by default, as in our batch-wise condition.[1] TGN also uses earlier test interactions for later predictions, with batching balancing computation and update frequency (Rossi et al., 2020). Computational batches do not always impose this history boundary. DyGLib models such as TGAT and DyGFormer retrieve neighbors strictly before each query timestamp, including earlier test interactions (Yu et al., 2023).[2] Repeating the event-wise evaluation with the saved weights gives nearly identical test scores.

Table 5: Test MRR (%) on TGB under event-wise and batch-200 updates. EdgeBank, Heuristic, Base3, and Ours share the listed schedule and candidates. Our scores are means and sample standard deviations over five runs. Neural reference scores retain their original protocols and are repeated for comparison. Bold marks the highest MRR in each row.

						Best neural	
Dataset	Updates	EdgeBank	Heuristic	Base3	Ours	MRR	Method
wiki	Event	64.40	82.06	73.26	**82.84** ±0.04	82.70	TPNet
	Batch 200	57.10	72.90	68.07	74.08 ±0.09	**82.70**	TPNet
uci	Event	32.40	**52.75**	35.22	51.57 ±0.32	24.50	TNCN
	Batch 200	22.16	37.49	28.22	**38.55** ±0.16	24.50	TNCN
enron	Event	15.60	84.66	48.65	**86.83** ±0.03	37.90	TNCN
	Batch 200	14.14	**54.93**	45.17	52.46 ±0.17	37.90	TNCN
subreddit	Event	59.22	74.19	74.25	**74.90** ±0.06	69.60	TNCN
	Batch 200	58.85	74.18	74.14	**74.30** ±0.04	69.60	TNCN
lastfm	Event	2.63	16.58	11.37	**32.84** ±0.18	15.60	TNCN
	Batch 200	2.57	15.74	11.32	**16.60** ±0.12	15.60	TNCN
review	Event	2.69	33.93	12.87	45.76 ±0.39	**52.10**	GraphMixer
	Batch 200	2.53	31.41	8.17	45.57 ±0.39	**52.10**	GraphMixer
coin	Event	58.31	81.10	77.71	82.09 ±0.06	**88.47**	CRAFT-R
	Batch 200	57.96	80.64	76.87	81.36 ±0.04	**88.47**	CRAFT-R
comment	Event	15.03	72.38	45.41	54.76 ±0.87	**91.72**	CRAFT
	Batch 200	14.94	72.07	44.93	54.58 ±0.86	**91.72**	CRAFT
flight	Event	38.71	88.92	79.34	89.99 ±0.04	**91.39**	CRAFT-R
	Batch 200	38.71	88.90	79.24	89.95 ±0.04	**91.39**	CRAFT-R

Matched counting baselines. Tables 5 and 6 report absolute scores under each update schedule. Under both schedules, our model exceeds EdgeBank and Base3 on all 16 datasets and the heuristic family on 14. The heuristic is stronger on `tgbl-uci` and `tgbl-comment` under event-wise updates, and on `tgbl-enron` and `tgbl-comment` under batch-wise updates. EdgeBank uses the TGB window rule under both schedules (Appendix A).

[1] EdgeBank code and TGB example.

[2] TGN evaluation, DyGLib evaluation, and neighbor sampling.

Table 6: Test MRR (%) on TGB-Seq under event-wise and batch-200 updates. Columns and conventions follow Table 5. Neural reference scores retain their original protocols.

						Best neural	
Dataset	Updates	EdgeBank	Heuristic	Base3	Ours	MRR	Method
GoogleLocal	Event	1.96	18.39	3.97	31.42 ±0.03	**62.88**	SGNN-HN
	Batch 200	1.96	18.39	3.73	29.00 ±0.01	**62.88**	SGNN-HN
Yelp	Event	9.77	30.81	11.99	53.21 ±0.06	**72.69**	CRAFT
	Batch 200	9.76	30.81	11.88	52.26 ±0.05	**72.69**	CRAFT
Taobao	Event	20.28	42.99	20.17	59.58 ±0.03	**70.68**	CRAFT
	Batch 200	20.27	42.99	20.16	59.57 ±0.03	**70.68**	CRAFT
ML-20M	Event	1.94	17.71	6.64	**39.00** ±0.10	35.91	CRAFT
	Batch 200	1.94	17.71	6.70	31.63 ±0.09	**35.91**	CRAFT
Flickr	Event	1.96	38.38	34.81	**62.68** ±0.07	62.34	CRAFT
	Batch 200	1.96	38.35	34.54	56.61 ±0.05	**62.34**	CRAFT
YouTube	Event	1.96	56.64	38.09	**65.27** ±0.05	59.64	SGNN-HN
	Batch 200	1.96	56.48	37.61	**63.89** ±0.03	59.64	SGNN-HN
WikiLink	Event	1.96	47.70	26.61	67.97 ±0.04	**75.48**	CRAFT
	Batch 200	1.96	47.67	24.93	58.01 ±0.02	**75.48**	CRAFT

With all counting methods evaluated in batches of 200, our model still achieves the highest MRR on 4 of 16 datasets: tgbl-uci, tgbl-subreddit, tgbl-lastfm, and YouTube, compared with 7 under event-wise updates.

Strict-timestamp sensitivity. Table 7 isolates the effect of excluding earlier events at the query timestamp. The largest reductions occur on tgbl-enron, WikiLink, Flickr, and YouTube. Configurations, weights, and candidates remain fixed, so the changes reflect the history available when each query is scored. Baselines were not evaluated under this schedule.

Table 7: Strict-timestamp sensitivity of our model. Scores are test MRR (%), with means and sample standard deviations over five runs. Changes from event-wise evaluation use unrounded scores paired by seed. Configurations, weights, and candidates are fixed.

Dataset	Event-wise	Strict timestamp	Change (points)
TGB			
wiki	82.84 ±0.04	82.84 ±0.04	0.00 ±0.00
uci	51.57 ±0.32	51.43 ±0.32	-0.14 ±0.00
enron	86.83 ±0.03	53.50 ±0.18	-33.33 ±0.16
subreddit	74.90 ±0.06	74.89 ±0.06	0.00 ±0.00
lastfm	32.84 ±0.18	32.78 ±0.18	-0.06 ±0.00
review	45.76 ±0.39	44.93 ±0.41	-0.83 ±0.06
coin	82.09 ±0.06	81.67 ±0.04	-0.42 ±0.03
comment	54.76 ±0.87	54.76 ±0.87	0.00 ±0.00
flight	89.99 ±0.04	88.94 ±0.12	-1.05 ±0.10
TGB-Seq			
GoogleLocal	31.42 ±0.03	31.42 ±0.03	0.00 ±0.00
Yelp	53.21 ±0.06	53.21 ±0.06	0.00 ±0.00
Taobao	59.58 ±0.03	59.58 ±0.03	0.00 ±0.00
ML-20M	39.00 ±0.10	39.00 ±0.09	0.00 ±0.00
Flickr	62.68 ±0.07	54.97 ±0.07	-7.71 ±0.05
YouTube	65.27 ±0.05	61.58 ±0.03	-3.69 ±0.05
WikiLink	67.97 ±0.04	56.25 ±0.04	-11.72 ±0.06