t-link, a simple and competitive method for Link Prediction in Temporal Graphs

Anonymous Author(s)

ABSTRACT

How to simply and accurately predict links in a temporal graph? Temporal graphs are powerful ways to model real world problems and many deep learning methods have been proposed recently to learn their representation and predict their evolution. These methods show good performance on various tasks but they are complex and simpler alternatives are lacking.

In this work we propose *t-link*, a new method for link prediction in temporal graph that is simple, fast and competitive. Compared to the state of the art, *t-link* is more than 100 times faster to train, 4 times faster at inference and on average 28% better in accuracy. Experiments performed on 4 benchmark datasets illustrate the benefits of our method.

KEYWORDS

machine learning, temporal graphs, link prediction

ACM Reference Format:

Anonymous Author(s). 2023. *t-link*, a simple and competitive method for Link Prediction in Temporal Graphs. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (KDD 2023).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Low dimensional representation learning is widely used in Machine Learning through self supervised and semi supervised methods. These representations capture the key features of the data which are further utilized in downstream tasks. These representation learning methods are employed in graph data with numerous applications including drug discovery, recommendation systems, knowledge graphs etc. Graph representation learning (GRL) includes static and temporal GRL. Contrary to its static counterpart, temporal GRL accounts for the evolution of data over time which is critical in a lot of applications where temporal patterns are significant [14]. For example, a common pattern in item recommendations is that similarity between items decreases over time (see figure 1).

Temporal GRL is powerful but still relatively recent and understudied compared to static GRL [14]. Naturally the baselines for temporal GRL are less mature than their static counterparts. We distinguish two categories of desirable baselines: strong baselines and simple baselines. Strong baselines are methods considered state-of-the-art (SOTA) for the task at hand. Simple baselines are methods with a high performance to complexity ratio. While the strong baselines for temporal GRL are emerging [14], [18] , **the simple baselines for temporal GRL are not clearly identified** and authors use static GRL baselines like graph auto encoders [6] or node2vec [4] that are relatively complex and ignore the temporal aspect of the problem. There is a need for simple baselines in temporal GRL.

The proper selection of baselines is a concern for the machine learning research community as some studies do not include strong baselines [10] or well optimized simple baselines [3], [1]. **Simple baselines are important**. They are important for researchers as they help understand a ML task and the relative value of new complex methods. They are important for practitioners who may find that they are better suited for industrial applications.

In this work we propose **t-link** a novel method for temporal GRL. The key contributions of our proposed method are:

- Simplicity: *t-link* embeds time in the graph structure with two simple steps, time discretization and pruning. Once the graph structure is updated, *t-link* learns temporal representations using a simple relational graph embeddings method.
- Speed: Because *t-link* uses a simple relational graph embeddings methods, it is 100 times faster to train and 4 times faster at inference than recent SOTA.
- Accuracy: We evaluated *t-link* across four SOTA benchmarks algorithms. To evaluate the impact of dataset size, number of items, number of interactions we use four dataset with varying characteristics. Our evaluation results demonstrate that *t-link* achieves 28% relative improvement compare to SOTA on two different metrics.

We present the background and related methods in Section 2. In Section 3 we present the details of our proposed method. In Section 4 we cover the experiments and evaluations conducted to demonstrate the effectiveness of *t-link*. Finally, Section 5 provides the key takeaways and conclusion.

2 BACKGROUND AND RELATED WORK

2.1 TransE

TransE is a simple method which models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities [2]. TransE is well known and widely used for GRL, we consider TransE a simple baseline for GRL.

TransE takes a graph in the form of relational triplets (source entity, relation, destination entity) formally represented by triplets of indexes $(i^s, i^r, i^d) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, and maps the triplets to an embedding space $\{(s, r, d) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n\}$.

Negative triplets are sampled during training by corrupting the source or destination node of positive triplets (positive triplets are the one present in the training data). In this work we follow

KDD 2023, August 06, 2023, Long Beach, California, United States

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

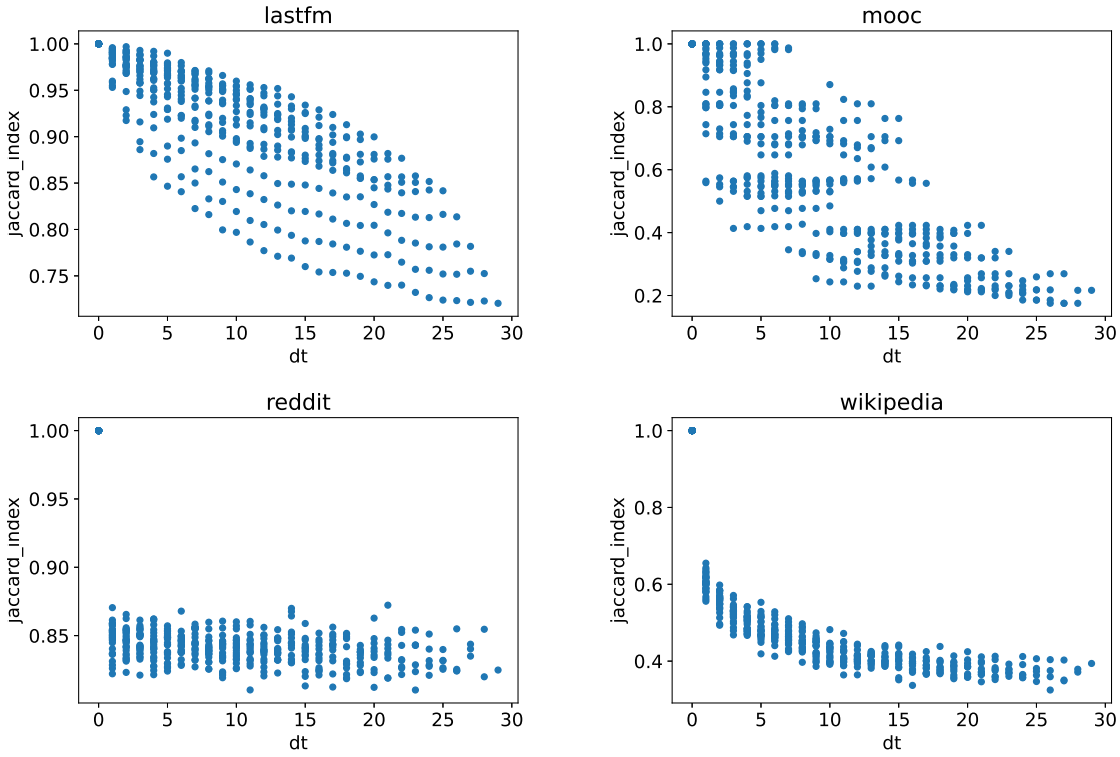

Figure 1: Graph data exhibits temporal patterns. Similarity between interacted items as a function of time on datasets LastFM, MOOC, Reddit and Wikipedia [8]. Interacted items are partitioned by time into buckets $I_q, q \in [0..29]$ of the same size. On the y axis we plot the set of values $\{J(I_q, I_{q+dt})|q \in [0..29 - dt]\}$ for each time delta $dt \in [0..29]$. $J(A, B) = |A \cap B|/|A \cup B|$ is the Jaccard index representing the similarity between 2 sets of items.

the implementation details of [20], in particular the use of the log sigmoid loss function:

$$L_{\gamma,p}(y, s, r, d) = \log(1 + \exp(y(||s + r - d||_p - \gamma)))$$

where $y \in \{1, -1\}$ indicates if the triplet is positive or negative, p is the order of the norm and γ is a threshold. Loss is minimized by gradient descent, pushing the distance $||s + r - d||_p$ below γ for positive samples and above γ for negative samples. Informally the objective of TransE is that $s + r \approx d$ if and only if the relation (s, r, d) is true.

The inference or scoring of a triplet is simply done by computing the distance function $||s+r-d||_p$. The time order of complexity is n, the embedding dimension. For top K link prediction we first define the prediction space Q of all triplets of interest (e.g. the triplets with a given source and relation but all possible destination nodes). Then we sort the triplets in Q by ascending distance $||s + r - d||_p$ and select the first K. The time order of complexity is $|Q| * (n + \log |Q|)$. In particular for predicting the top K items in I for each user in U, the time order of complexity is $|U| * |I| * (n + \log |I|)$.

2.2 Temporal TransE

TransE being a well established method for GRL, several attempts have been made to adapt TransE to temporal graphs. In [5] the authors modify TransE to take into account the order of relations like "born in" and "died in". The method is effective but does not address the more general case of the effect of time on a single relation and requires knowledge of the order of relations. In [9] the authors study several modifications of TransE for temporal validity and link prediction but none of those modifications achieve satisfactory results. One of their proposals (using a synthetic relation $r : t$ for each relation r and timestamp t) is similar to our method but lacks the discretization of time which is key to make the method scalable.

2.3 Temporal Graphs Neural Networks

In the more general context of attributed temporal graphs recent work has leveraged on sequential models and graph neighborhood aggregation with attention. In [8] the authors propose a method to jointly learn and predict the trajectories of users and items in temporal bipartite graph. User and item embeddings are updated alternatively using recurrent neural networks at each interaction involving them. Then a time projection is applied to project the user embedding and the predicted item embedding from the last interaction to the desired time. In [18] the authors propose a temporal graph attention layer to aggregate temporal-topological neighborhood features as well as to learn the time-feature interactions. In particular they use Fourier decomposition to encode continuous time under the assumption that "relative time span, rather than the

absolute value of time, reveals critical temporal information". [14] combines and extends the two methods above into a general framework for temporal graphs networks (TGN). Note that in addition to the Fourier time encoding used in [18], [14] also incorporates simple inductive biases like sampling from the most recent neighbors. [14] introduces some clarity by presenting methods above as examples of TGN but TGN itself exhibits significant complexity with a memory module, memory updater, embedding module, message aggregator and message function.

3 PROPOSED METHOD

The idea of our proposal is to prepare the temporal graph data so that temporal information becomes implicit in the graph structure and non temporal relational graph methods can be applied. We chose TransE [2] for its simplicity and performance, but our method can be used in conjunction with all relational graph methods like RESCAL [13], DistMult [19], TransR [11], ComplEx [17], RotatE [15] etc....

Below we describe the two graph preparation steps, discretization and pruning, and the training and inference procedure.

A temporal graph can be seen as a set of quadruplets (i_s, i_r, i_d, t) which are simply the index triplets described in section 2.1 augmented by timestamps $t \in T \subset \mathbb{R}$.

Discretization: In this step, timestamps $t \in T$ are mapped to a category by the function $c_q(t) \in [0, 1, ..., C_q - 1]$. Where, C_q is the number of categories and q a hyper-parameter controlling the discretization. The category is then concatenated to the relation index to form a temporal relation index $i_r||c_q(t)$. After discretization the graph data is no longer explicitly temporal and can be seen as a set of triplets:

$$(i_s, i_r, i_d, t) \rightarrow (i_s, i_r||c_q(t), i_d)$$

The discretization function $c_q(t)$ has two modes depending on the value of q. If $0 < q < 1$, time is discretized in two categories defined by t_q, the quantile at q of the set of timestamps T:

$$c_q(t) = 1[t > t_q], C_q = 2$$

If $q \in \mathbb{N}^+$, time is discretized into q iso-quantile categories:

$$c_q(t) = j | t \in T_j, C_q = q$$

where the set $\{T_j, j \in [0, ..., q-1]\}$ is the iso-quantile partition of timestamps T into q intervals. For example if $q = 4$ then $T_0 = \{t \in T | t_0 \leq t < t_{0.25}\}$ and $T_3 = \{t \in T | t_{0.75} \leq t \leq t_1\}$.

Note than if $q = 1$ then $c_q(t) = 0, \forall t$ and discretization has no effect on the data. Also note that the oldest and most recent timestamps are characterized respectively by $c_q(t) = 0$ and $c_q(t) = C_q - 1$.

With this modification *t-link* learns distinct embeddings for each time bucketed relation $i_r||c_q(t)$. This allows *t-link* to focus on recent interactions at inference time without losing information about the past.

Pruning: In this step, the triplets are read in decreasing order of timestamps and each triplet is attributed a rank based on the order of appearance of the destination node i_d for a given source i_s and relation i_r. The first appearance (most recent) is ranked 1, second appearance is ranked 2, such that all appearances are ranked from most recent to least recent. Next the triplets where $rank > k_r$ are removed from the data. k_r is the user defined pruning threshold that is used for limiting the length of historical interactions for a node on a given relation. Pruning is a more drastic way to focus on recent destination nodes by removing the less recent ones. Note that when $k_r = \infty$ pruning has no effect on the data.

The time order of complexity of discretization and pruning is m if the dataset is pre-sorted by timestamps, $m \log(m)$ otherwise, m being the number of triplets in the dataset.

NB: if discretization and pruning are deactivated then *t-link* becomes TransE. The contribution of each step is shown in section 4.7.

Training: model training is done on the triplets using the TransE procedure described in section 2.1. For each original relation the model will learn C_q temporal relations from the least recent to the most recent.

Inference: for future link prediction we apply the top K prediction procedure described in section 2.1 with a restriction on the given source node i_s and most recent relation $i_r||C_q - 1$. The intuition is that future interactions are more similar to recent interactions compared to past interactions, so we search for probable future interactions in the neighborhood of recent ones. The same reasoning can be applied to past interactions, for example in the context of knowledge completion we can be interested in predicting links in the past and would use a different time bucket for inference. In general for a given source node i_s, relation i_r and time t we can define the projected destination embedding as:

$$\widehat{d(t)} = s + r_{i_r||c_q(t)}$$

where $r_{i_r||c_q(t)}$ is the learned embedding of temporal relation $i_r||c_q(t)$. For extrapolation ($t < \min(T)$ or $t > \max(T)$) we set respectively $t = \min(T)$ or $t = \max(T)$, T being the set of training timestamps.

Reproducibility: the code for *t-link* including hyper-parameters tuning can be found here[1].

4 EXPERIMENTS

Here we report experiments to answer the following questions:

- Q1. Accuracy: How accurate is *t-link* for link prediction?
- Q2. Speed: How fast is *t-link* in terms of training and inference?
- Q3. Ablation study: What are the contributions of discretization and pruning to *t-link* accuracy?

We start by describing the datasets, evaluation method, metrics and baselines.

4.1 Datasets

We use the datasets provided by [8] namely LastFM, MOOC, Reddit and Wikipedia. These datasets represent bipartite graphs of users and items with varying level of repetition in the interactions and interaction features. Nodes do not have features. If present, edges features are continuous. Note that our method *t-link* will ignore the edge features and consider all edges representing the same relation.

[1] to be released

In table 1 we report some characteristics of the datasets. We report a local and a global repetition rate: the local repetition rate looks at how many times an item is directly followed by the same item. The global repetition rate looks at the number of interactions and unique items globally without order. We see for example that LastFM has a low local repetition rate (users rarely listen to the same song twice in a row) but a high global repetition rate (users come back to the same songs eventually). For users editing Wikipedia pages the local repetition exceeds global repetition (users tend to iterate many times in a row when modifying an article but then do not come back to it). We also report a sparcity index, which is the number of users multiplied by the number of items divided by the number of interactions.

The LastFM dataset stands out by its large time span and absence of edge features. In [8] the authors mention a time span of 1 month but we observe a time span of about 4 years. The MOOC dataset stands out in terms of number of items (around 100 items compared to around 1000 for other dataset). Because of the low number of items the MOOC dataset is less relevant for link prediction and other authors used it only for node classification. We include it for completeness. LastFM and MOOC have an order of magnitude lower sparsity index compared to Reddit and Wikipedia.

In figure 1 we see that the datasets exhibit various degrees of auto-correlation in the sense of Jaccard similarity between items in different time buckets.

4.2 Evaluation method

We use the same time based data split as in [14] and [18]: edges are sorted by ascending timestamps, the first 70% are used for training, the next 15% for validation, the final 15% for testing. Instead of the sampled evaluation used in [14] and [18] we prefer a global and exact evaluation based on the top K predictions for each user in the dataset (K=100). There are two main reasons for this. First, global top K evaluation is more reliable, as sampled evaluation introduce biases that depend on the metric, the sample size and the recommendation algorithm [7]. Second, global top K evaluation is more simple, auditable and reproducible, as it can be computed offline from the list of top K predictions per user.

Another important difference in our experimental set-up is that it is "offline": the model is trained on the training set, uses the validation set to optimize hyper-parameters and produces top K recommendations that are compared to the test set. This is different from the "online" set-up in [14] and [18] where the model parameters and/or memory are updated by validation and test batches of edges so the model is only asked to score the next batch in the immediate future and compare with scores given to negative samples. In the offline setup the model is asked for each source node to predict the top K destination nodes in a "distant" future (test set).

To ensure fair comparison between methods we add a global top K prediction script to the TGN framework provided by [14]. The script loads the pretrained model and computes embeddings for all source nodes and destination nodes at time $(max(T) + min(T)/2$ where T is the set of timestamps for the test set. Then for each source node (user) the TGN scoring function is applied to compute link probabilities for all possible destination node (item) and the top K most likely items are persisted. The same code is used to compute metrics for all methods based on persisted predictions.

t-link is optimized using grid search, varying the discretization parameter q, the pruning threshold k (there is only one type of relation in the data so only one k), the γ threshold and the learning rate. For each dataset we retain the hyper-parameters with the best validation MRR and report the test MRR and test MAP.

For the baselines we retain the hyper-parameters used in [14].

4.3 Metrics

Once the recommendations are produced, they are evaluated on the test set with the standard metrics below.

Reciprocal Rank: Each user in the training set gets 100 item recommendations ranked by score from 1 (most likely) to 100 (less likely). The rank R is the lowest rank among relevant recommendations (recommendations that appear in the evaluation set). If there are no relevant recommendations in the top 100 we set the rank arbitrary to 1000. Reciprocal rank is $1/R$.

Average Precision: For each relevant recommendation we consider the list of recommendations up to this recommendation (included), and compute the precision of the list (number of relevant recommendations / length of the list), then all the precision values are averaged. If there are no relevant recommendations in the top 100 we set the average precision to 0. Note that if only one recommendation is relevant in the list then average precision is the same as reciprocal rank.

Metrics are averaged across all training users who appear in the evaluation set, and referred as MRR (mean reciprocal rank) and MAP (mean average precision).

4.4 Baselines

Our strong baselines are TGN [14], TGAT [18], JODIE [8] and DyRep [16]. We use the code of TGN to run these strong baselines as it is a general framework that supersedes them. As stated above TGN does not provide a global top K prediction script so we provide our own. The code for reproduction can be found here[2].

Our simple baseline is TransE [2], it can be reproduced with the *t-link* code or directly from the DGL-KE [20] code [3].

4.5 Q1 - Accuracy

Tables 2 illustrates the performance of *t-link* on future link prediction. We see that *t-link* outperforms complex methods on both metrics on the LastFM, Reddit and Wikipedia datasets, with a minimum absolute improvement of 10% (MRR on Reddit). TGN [14] outperforms on both metrics on the MOOC dataset. Averaged across datasets and metrics, the relative accuracy improvement of *t-link* over TGN is 28%.

Plain TransE also outperforms TGN and other complex methods on LastFM, Reddit and Wikipedia in terms of MAP, indicating that non temporal patterns of the data are important and not always well captured by complex temporal graph methods. By contrast *t-link* improves upon TransE on all metrics and datasets.

We hypothesize that our method outperforms on 3 out of 4 datasets because it is focusing on the dominant data characteristics

[2] to be released
[3] https://github.com/awslabs/dgl-ke

Table 1: Datasets statistics

	lastfm	mooc	reddit	wikipedia
# users	980	7,047	10,000	8,227
# items	1,000	97	984	1,000
# interactions	1,293,103	411,749	672,447	157,474
sparcity	0.76	1.66	14.6	52.2
# edge features	0	4	172	172
time span (days)	1,587	30	31	31
global repetition (%)	67	46	85	40
local repetition (%)	11	17	62	48
interaction type	song listening	mooc interacting	posting	editing

Table 2: Metrics for top 100 future predictions (%). Bold fonts for best metric, underline for second best.

	lastfm		mooc		reddit		wikipedia	
	mrr	map	mrr	map	mrr	map	mrr	map
dyrep	9.9	7.1	21.9	16.5	1.2	0.9	0.9	0.8
jodie	4.1	4.9	50.3	37.3	25.9	23.7	9.5	8.2
tgat	28.9	15.0	45.2	30.9	74.9	62.3	29.6	27.1
tgn	36.7	21.6	**66.3**	**46.6**	73.8	62.4	38.0	34.0
transe	45.7	30.8	22.8	20.4	71.3	63.4	55.8	53.1
t-link	**54.7**	**33.0**	56.0	35.4	**85.5**	**76.0**	**60.9**	**57.5**

and patterns by design and therefore achieves a good fit without a very large or dense dataset. Dominant patterns are likely to be: users who interact on the same items are similar (this pattern should be well captured by TransE), recent interactions are more predictive than old interactions (this pattern is targeted by TGP). Other patterns can be: users who interact with item A tend to interact with item B immediately afterwards, users who interact with item A in a particular way tend to interact again with item A. TGN is using edge features to build update messages and a recurrent network to update node memories so it has the ability to learn these other patterns but may require more data to effectively do so. The performance of TGN on the MOOC dataset may be linked to the lower number of items (10 times less than other datasets) and low sparcity which is more favorable to a complex model.

4.6 Q2 - Speed

Table 3 shows the training and inference time of all methods on an AWS instance ml.p3.2xlarge (single GPU NVIDIA Tesla V100). We see that our method is more than 100 times faster to train and 4 times faster for inference (top 100 predictions) compared to the SOTA method TGN.

4.7 Q3 - Ablation study

From the grid search results we identify the experiments with best validation MRR for each ablation configuration (TransE only, TransE with pruning, TransE with discretization and TransE with discretization and pruning) and report the test MRR in figure 2.

Table 3: Training and inference time (s) for dataset LastFM on AWS instance ml.p3.2xlarge (single GPU NVIDIA Tesla V100)

	training	inference
dyrep	1474.1	2.73
jodie	972.4	2.72
tgat	20852.0	3.03
tgn	3250.9	2.73
t-link	**30.9**	**0.67**

We also compute SHAP [12] style additive contributions of each component (average increase of MRR by application of component first or second), these contributions are visualized in figure 3.

We see that pruning is useful on its own but less significantly when applied in conjunction with discretization. *t-link* with discretization alone outperforms the current SOTA on LastFM, Reddit and Wikipedia. A reader interested in the simplest temporal GRL baseline can restrict *t-link* to discretization. Discretization also has the benefit to be controlled by a single hyper-parameter, whereas pruning is controlled by a hyper-parameter by relation type.

While the accuracy improvement of *t-link* over TransE is significant and positive on all datasets, we observe a large variation in the improvement size: on the MOOC dataset the MRR is increased in absolute by 33%, on the Wikipedia dataset by 5%.

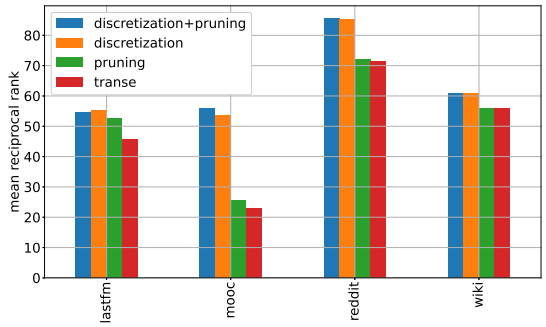

Figure 2: MRR for all *t-link* variants and datasets (from left to right: t-link with discretization and pruning, t-link with discretization, t-link with pruning, TransE)

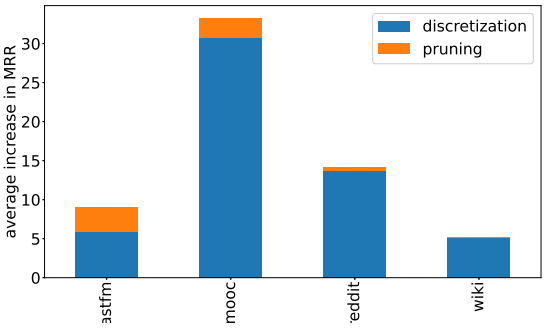

Figure 3: Contribution of discretization and pruning in *t-link*, per dataset

5 CONCLUSION

We presented *t-link*, which addresses the link prediction problem in temporal graphs, using temporal graph preparation steps. The main idea is to embed time in the graph structure so that non temporal relational graph embedding methods can learn recency preferences from the graph structure.

The advantages of our method are:

- Simplicity: *t-link* embeds time in the graph structure with two simple steps, discretization and pruning
- Speed: *t-link* is 100 times faster to train and 4 times faster at inference than recent SOTA
- Accuracy: averaged across 4 benchmark datasets and 2 metrics, *t-link* relative improvement over the recent SOTA is 28%

The main contributor to *t-link* is time discretization which consists in replacing each timestamped relation by the concatenation of the relation and the time category. For researchers and practitioners using temporal graphs for link prediction, we recommend benchmarking their methods with global top K evaluation and consider very simple baselines like TransE and *t-link*.

Future work may include studying the application of *t-link* to online, next batch prediction. We may also investigate to understand more precisely what datasets and patterns are favorable to our simple method vs more complex temporal graphs methods.

REFERENCES

[1] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. 2022. Top-N Recommendation Algorithms: A Quest for the State-of-the-Art. In *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization* (Barcelona, Spain) *(UMAP '22)*. Association for Computing Machinery, New York, NY, USA, 121–131. https://doi.org/10.1145/3503252.3531292

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[3] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 101–109. https://doi.org/10.1145/3298689.3347058

[4] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[5] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards Time-Aware Knowledge Graph Completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, 1715–1724. https://aclanthology.org/C16-1161

[6] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. https://doi.org/10.48550/ARXIV.1611.07308

[7] Walid Krichene and Steffen Rendle. 2022. On Sampled Metrics for Item Recommendation. *Commun. ACM* 65, 7 (jun 2022), 75–83. https://doi.org/10.1145/3535335

[8] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[9] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving Validity Time in Knowledge Graph. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1771–1776. https://doi.org/10.1145/3184558.3191639

[10] Jimmy Lin. 2019. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (jan 2019), 40–51. https://doi.org/10.1145/3308774.3308781

[11] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of the AAAI Conference on Artificial Intelligence* 29, 1 (Feb. 2015). https://doi.org/10.1609/aaai.v29i1.9491

[12] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[13] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (Bellevue, Washington, USA) *(ICML'11)*. Omnipress, Madison, WI, USA, 809–816.

[14] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[15] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. https://doi.org/10.48550/ARXIV.1902.10197

[16] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyePrhR5KX

[17] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 2071–2080. https://proceedings.mlr.press/v48/trouillon16.html

[18] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJeW1yHYwH

[19] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. https://doi.org/10.48550/ARXIV.1412.6575

[20] Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. 2020. DGL-KE: Training Knowledge Graph Embeddings at Scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 739–748.