Supplementary Information

Training details. We developed DURENDAL using Pytorch Geometric (PyG) [1]. We use the implementation available in PyG for GAT, HAN, and HGT. For EvolveGCN, GCRN-GRU, and HetEvolveGCN, we use the implementation available in Pytorch Geometric Temporal [2]. We ran our experiments on NVIDIA Corporation GP107GL [Quadro P400]. In all our experiments, we use the Adam [3] optimizer. This choice was made according to some prior works on GNN architecture for temporal heterogeneous networks [4, 5, 6, 7]. We adopt the live-update setting to train and evaluate the models, wherein we engage in incremental training and assess their performance across all the available snapshots. With respect to each snapshot, a random selection of 20% of edges is employed to establish the early-stopping condition (validation set), while the remaining 80% are utilized as the training set. The edges contained within the subsequent snapshot constitute the test set. Consistently, we apply identical dataset divisions and training procedures across all the methods. Hyperparameters are tuned by optimizing the AUPRC on the validation set, and the model parameters are randomly initialized. The hyperparameter search spaces are as follows: learning rate {0.1, 0.01, 0.001}, L2 weight-decay {5e-1, 5e-2, 5e-3}, number of hidden layers {1, 2}, representation dimension {32, 64, 128, 256}. For DURENDAL, we tested also three different message-passing operators: GAT, SAGE [8], and the operator from the work by Morris *et al.* [9] (GraphConv).

Model architectures. We tested DURENDAL and all the other baselines using either one or two graph message-passing hidden layers. We do not test architecture with more than two graph message-passing layers to avoid the over-smoothing problem [10]. We define and test two different configurations for the number of hidden neurons: the first has 64 and 32 hidden neurons and the second has 256 and 128. DURENDAL, GAT, and HAN achieve their best performances using 2 layers, while the other models use only one hidden layer. All the models reach better performance with the highest number of hidden neurons among the considered dimensions for each layer. For DURENDAL, we report in Table 1 the best configuration for the update modules for each dataset. The results presented in the paper are obtained using a DURENDAL model with GraphConv as the message-passing operator and semantic-level attention mechanism [6] to aggregate partial node representations.

Table 1: Best embedding update module of DURENDAL models for future link prediction over the four THNs datasets.

Dataset	Update module
GDELT18	GRU
ICEWS18	GRU
TaobaoTH	Weighted average
SteemitTH	ConcatMLP

Resources and computational cost. Table 2 reports the hardware specifics of the machine on which we run algorithms for data gathering, preprocessing, and all the experiments described in the paper. Overall, computing all the experiments with all the baselines and a single configuration of hyperparameters takes about 1 day and a half. We describe the amount of computational time for individual experiments in Table 3. We report for each dataset the overall computational time of the pipeline from the data loading until the output of the prediction for each candidate model, considering one configuration of hyperparameters. A crucial part of the work is related to the data gathering and preprocessing but we do not report their computational time since we provide the obtained datasets. All the reported computational times are provided approximately.

Data gathering and preprocessing. Since we can not release the complete dataset related to `SteemitTH` (see Data-related concerns in Section 6), we briefly summarize how to collect and preprocess data to obtain it in the following steps:

1. Collect data from June 3, 2016, to February 2, 2017, using the Steemit API [11]. Consider the first 3 months as the initial training snapshot and the following months as subsequent snapshots. "follow" interactions are available in `custom_json` operations, transactions are available in `transfer` operations, votes in `vote` operations. Posts and comments written by users are available in `comment` operations.

Resource	Description
CPU	Intel Core i9-9820X CPU @ 3.30GHz x 20
GPU	NVIDIA Corporation GP107GL [Quadro P400]
RAM	64GB
Disk	256 GB

Table 2: The hardware specifications of the machine utilized for executing algorithms involved in data gathering, preprocessing, and all the experiments detailed in the paper.

Experiment	Running time
GDELT18	10min
ICEWS18	10min
SteemitTH	30min
TaobaoTH	20h
Effectiveness update-scheme	8h
Effectiveness model-design	6h

Table 3: Approximate computational time for individual experiments

2. Construct a `HeteroData` object with a single node type and four relation types: “follow”, “vote”, “comment” and “transaction”. To construct the edge list related to each relation, you can refer to SteemOps [12] which describes in detail the schema of each operation and which field contains the ids of the nodes involved in each interaction.
3. For the textual content X, we use a pre-trained SBERT language model [13] to obtain points in the Euclidean space. For each time interval t, we call $D_{(u,t)}$ the collection of documents (posts and comments) posted by user u during time interval t. To obtain the initial node features $X_{(u,t)}$ of u at time t, we average its document embeddings, that is $X_{(u,t)} = \frac{1}{|D_{(u,t)}|} \sum_{d \in D_{(u,t)}} \mathrm{SBERT(d)}$ using the element-wise sum. Users with no published textual content - missing node features - have a zeros vector as initial features.
4. Repeat steps 2-3 for each snapshot.

To process textual content we use the `all-MiniLM-L6-v2` SBERT model. We choose this model because *i*) is trained on all available training data (more than 1 billion training pairs), *ii*) is designed as a general purpose model, and *iii*) is five times faster than the best SBERT model but still offers good quality[1].

For `GDELT18`, `ICEWS18`, and `TaobaoTH`, we download the source data from the PyG library [1]. We release the code to compute data preprocessing and obtain the graph snapshots representation to train and test DURENDAL. For further details, you can inspect the annotated code in the GitHub repository.[2]

References

[1] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[2] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573, 2021.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[1] `https://www.sbert.net/docs/pretrained_models.html`, May 2023

[2] `placeholder`, see the additional material.

[4] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6669–6683, Online, November 2020. Association for Computational Linguistics.

[5] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[6] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, page 2022–2032, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, page 2704–2710, New York, NY, USA, 2020. Association for Computing Machinery.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[9] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, Jul. 2019.

[10] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3438–3445, Apr. 2020.

[11] Steemit developer documentation. Broadcast Ops, 2021.

[12] Chao Li, Balaji Palanisamy, Runhua Xu, Jinlai Xu, and Jingzhe Wang. SteemOps. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. ACM, apr 2021.

[13] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.