Recommender System Prediction with Dynamic Graph Representation Learning: More Flexible and More Robust

Amine MOHABEDDINE, Leshanshui YANG, Sébastien ADAM, Clément CHATELAIN

Université de Rouen Normandie

1 Introduction

Predicting potential relations between network nodes commonly known as link prediction, has been a real challenge in recent years. Research effort has been predominantly focused on link prediction on static graphs. However, real world problems cannot be properly modeled by a static graph due to their dynamic nature, characterized by the emergence and disappearance of nodes and links over time. Consequently, the field of link prediction in dynamic graphs drawn increasing attention, as it offers a more precise comprehension of the evolving nature of real-world networks. In recent years, Dynamic Graph representations have become inescapable for modeling dynamic systems, incorporating both topological and temporal information. We can find these Dynamic graph representations in different domains such as social network prediction, Recommender Systems, traffic forecasting, and cybersecurity. The rise of dynamic graph representation has given rise to the field of dynamic graph learning which aim to develop models that can make predictions or learn patterns in the evolving graph structure. At the forefront of this research domain stands the Dynamic Graph Neural Network (DGNN), currently regarded as the pinnacle approach.

In our work, we are aiming to produce link prediction models that are more flexible and more robust. We are going to address two main problematic. The first one, is the capability of predicting links for nodes that have not been seen during the learning phase, also known as cold start. We are using a model called Node2Vec [2] to enhance new node embeddings. It is a model based on random walks exploring neighborhoods and learning richer representations. The second problematic is the behavior of link prediction models when adding a random proportion of noise edges in learning process. This simulates, automated or abnormal requests in a network. Are these requests affecting negatively the performance of link prediction models? We are going to answer this question in this work.

This report is organized as follows. In this first section we have introduced link prediction and talked about Recommender Systems which are directly related to each other. In the second section, we are going to provide some definitions on Recommender Systems with fundamental notions on dynamic graphs and possible prediction tasks for link prediction models. The third section is dedicated to the literature review. In the fourth section, we are giving a detailed presentation of Euler model for link prediction, which is going to be testing in different settings in the rest of this work. The fifth section is the main contribution of this work, We are exploring different approaches to help with the robustness and flexibility of link prediction models for Discrete Time Dynamic Graphs. In the first approach, we are addressing the cold start problem on Euler model for link prediction. We are going to explore embeddings initialization for unseen node using a node2vec model. In the second approach, we are injecting negative sample of edges with different proportions. And then observe the behavior of Euler model to these addition of edges. In the sixth section, we are presenting the experimental setup as well as the different experimental results presented in this paper. In the seventh and final section, we conclude our work by summarizing the obtained results and providing some future perspectives.

In: M. Meder, A. Rapp, T. Plumbaum, and F. Hopfgartner (eds.): Proceedings of the Data-Driven Gamification Design Workshop, Tampere, Finland, 20-September-2017, published at http://ceur-ws.org

2 Definitions

2.1 Recommender System

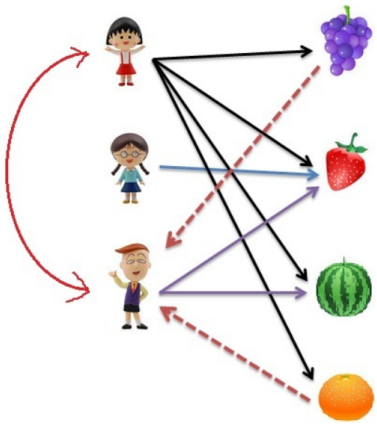


Figure 1: Recommender Systems based on previous interactions

Recommender Systems (RS) are algorithms aimed at suggesting to a given user relevant items or users sharing the same interests. To perform such suggestions, the RS have to predict future interests of the user based on previous interactions as shown in figure 1. RS can be applied in many domains: video streaming such as Youtube and Netflix, audio streaming like Spotify and Deezer, online stores, social media and many other platforms. There exists many approaches to tackle RS such as reinforcement learning methods or Collaborative Filtering methods (predict user preferences given the patterns of interactions among users and items). In this document we are exclusively addressing RS with Deep Learning-Based Approaches on dynamic graphs.

2.2 Dynamic Graphs

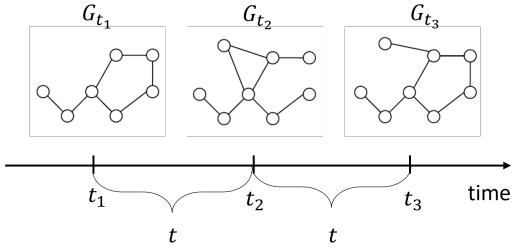
Dynamic graphs are graphs capable of handling time-varying topology and sometimes attributes evolution. There exists many terms used in the literature to describe such graphs : temporal graphs, dynamic graphs, dynamic networks, edge/node evolving graphs. For the rest of this document we will be referring to these graphs as Dynamic Graphs (DG). Before, understanding how dynamic graphs are modeled we need a brief overview of the static graph modeling.

2.2.1 Static Graph modeling

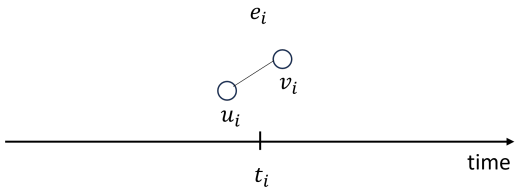
A static graph G is defined by a tuple $G = (V, E, \mu, \xi)$ where V is the nodes set, E is the edges set, μ and ξ are respectively a nodes attribute and edges attribute functions. This attributes map to each node v (edge e) attributes $\mu(v) \in U$ ($\xi(e) \in \Xi$). U and Ξ are respectively the set of possible nodes and edges attributes.

2.2.2 Dynamic Graph modeling

Dynamic graphs are defined as graphs that model topological and temporal information, in other terms, the graph will change over time by having nodes/edges appearing and disappearing, and node/edge attributes evolving. With this definition, one can encounter multiple configurations. Two main configurations are widely used: Discrete Time Dynamic Graphs (DTDG) and Continues Time Dynamic Graphs (CTDG) as shown in Figure 2.



(a) DTDG representation



(b) CTDG representation

Figure 2: Difference between discrete and continuous time dynamic graphs

Discrete Time Dynamic Graphs: Let $G = (\mathcal{V}, \mathcal{E})$ be a DTDG. The changes in the topology of G operate over a discrete time axis. Each time of this discrete axis is called a timestamp. The DTDG is characterized by the fact that the difference between two successive timestamps t_i and t_{i+1} is a fixed period. These properties allow the DTDG to be modeled as a sequence of static graphs. At each timestamp $t \in \{t_1, t_2, ..., T\}$ the DTDG G is modeled with a static graph $G_t = (V_t, E_t)$ also called snapshot where $V_t \subseteq \mathcal{V}$ and $E_t \subseteq \mathcal{E}$ for $t \in \{t_1, t_2, ..., T\}$.

Continued Time Dynamic Graphs: Let $G = (\mathcal{V}, \mathcal{E})$ be a CTDG. unlike a DTDG the changes in the topology of G are performed in a continuous time axis. The notion of snapshot should not be used in CTDG representation (infinite possible snapshots in a given time interval, where a whole snapshot will contain a single change in the topology of the graph). Thus, CTDG are usually modeled using a sequence of events. An event $e_i = (u_i, v_i, t_i)$ is an interaction between two nodes u_i and v_i at a given time $t_i \in [0, T]$, this could be a link appearance of disappearance between u_i and

v_i or an attribute change.

2.3 Predictive tasks

There exist many possible predictions that can be produced by a link prediction model on a dynamic graph. In what follows, we are focusing on two main predictive tasks, as well as, some real-life applications for both tasks.

2.3.1 Link prediction

Given a DG containing temporal information between times t_0 and t_n. Link prediction consists in predicting the existence or not of a future connection between two nodes at timestamp t_{n+k} with $k > 0$. link prediction is usually present in many applications such as recommender system for predicting users' future interests in general.

2.3.2 Link detection

Given a DG containing temporal information between times t_0 and t_n. Link detection consists in detecting abnormal links at time t_n. This approach is widely used in fraud detection and cybersecurity.

3 Literature Review

In [10], the authors proposed a survey on graph representation learning on dynamic graphs. In the first part of their work, the authors proposed definitions on DG representations as well as learning on these representations using different learning modes: transductive and inductive learning. Moreover, they summarized different predictive tasks along with real-life applications. In the second part, the authors have introduced vectorial representations (embeddings) for dynamic graphs, detailing the encoding of topological information related to each predictive task and each representation of DG. The authors provided detailed explanation on the sequential representation of hidden states for the temporal encoding. The authors explored the models used in the literature combining Graph Neural Network (GNN) and recurrent Neural Network (RNN) for respectively extract topological and temporal encoding. In the last part of their work, the authors have introduced guidelines for designing DGNN, including the definition of inputs, outputs, and predictive task, the selection of the right time encoding approach, the design of the Neural Network structure, and the optimisation of the DGNN model.

In [4], authors introduced their model called JODIE for link prediction on RS with CTDG. Their model is based on a sequence on events or interactions between users and items, and initial embeddings for both users and items. From these elements, the model will learn an embedding trajectory for users and items using two RNN: RNN_u and RNN_i for respectively updating user and item embeddings. To predict future interests of a given user u at time $t + \Delta$, where t was the time of the last interaction of the user u, the authors have introduced a projection operator $p(u, \Delta)$. This projection operator emulates the fact that the state of the user may change over time, so it will project the embedding of the user u before producing the prediction (the larger Δ is, the more the projected vector differs from the initial embedding vector).

King et. all [3], have proposed a temporal link prediction model called Euler for detecting network lateral movements on dynamic graphs. Their model combines Graph Convolution Networks (GCN) for topological encoding and RNNs for temporal encoding. Their main contribution consists in proposing a scalable model by training a different GCNs at each timestamp. The authors compared the performance of their model to other models on link prediction and link detection tasks using different datasets.

In [1], authors proposed a link prediction model E-LSTM-D (for Encoder-LSTM-Decoder) on DTDG. The model is using an auto-encoder architecture for encoding topological embedings first. These embedding are considered as input for a LSTM model which will encode temporal embeddings. Then, temporal representations are decoded to produce predictions using the decoder of the auto-encoder architecture. The model is using a weighted MSE Loss giving more weights to positive edges (existing connections) which are way less represented due to the sparsity of the adjacency matrix. Finally, they proposed an experimental comparison between their model and state of the art models.

In [6] authors have proposed a link prediction model with DTDG. Their model uses GCN layers to encode both topological and temporal embeddings. They use Evolving Graph Convolution Units (EGCU) in their convolution layers. These EGCUs perform graph convolutions with dynamic parameters. These dynamic parameters evolve dynamically in time using a recurrent neural network. To update the parameters of convolution layer at level l, the authors presented two different approaches that use recurrent neural network. The first one consists in using a GRU neural network with two inputs: the parameters of the layer $l - 1$ at time t, and the parameters of the layer l at time $t - 1$ (output of the previous GRU cell). In the second approach, they are using an LSTM with the

parameters of the layer l at time $t-1$ as the only input. The last part of their work was dedicated to a comparative experimentation using synthetic dataset as well as well as other datasets such as Bitcoin OTC, Reddit Hyperlink Network, or Autonomous systems.

4 Euler model for link prediction

In this section, we are going to give a detailed overview of Euler model for link prediction, which we are going to consider for the rest of this work.

4.1 Inputs

The Euler model is considering a DTDG $G = (\mathcal{V}, \mathcal{E})$ as input. So, for each timestamp t we are having a snapshot $G_t = (V_t, E_t)$ with $V_t \subseteq \mathcal{V}$ and $E_t \subseteq \mathcal{E}$ for $t \in \{1, ..., T\}$. The model is not considering node or edge attributes.

4.2 Encoder

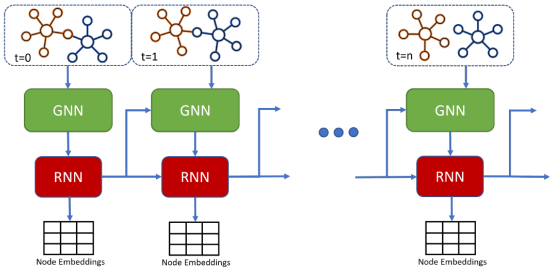


Figure 3: Euler model for link prediction on dynamic graphs architecture. This figure illustrates the encoding of topological embeddings with GCN and temporal embeddings with RNN.

4.2.1 Topological Embeddings

Euler uses GCN convolution layers to encode topological embedding. For each timestamp t a different GNN_t composed of two GCN convolution layers is trained to learn the topological representation from G_t. The output of each layer is calculated by:

$$H_t^{(l+1)} = \sigma(\tilde{D_t}^{-\frac{1}{2}} \tilde{A}_t \tilde{D_t}^{-\frac{1}{2}} W_t^{(l)})$$

- $H_t^{(l)}$ is the hidden state of the layer l at time t.
- $W_t^{(l)}$ are the parameters of the layer l
- $\tilde{A} = A_t + I$ where A_t is the adjacency matrix of G_t and I the identity matrix.
- $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$
- Given that there is no nodes features, Euler considers $H^{(0)}$ as a one hot representation.

4.2.2 Temporal Embeddings

Temporal embeddings are encoded with an RNN. the RNN takes as input the sequence of hidden states $H_t^{(l)}$ provided by GNN_t at each timestamp t (l is considered here as the last layer of the GCN). The output of each RNN cell is computed by:

$$h_t = W_h h_{t-1} + W_H H_t^{(l)}$$

The final node embeddings are denoted by:

$$\begin{aligned} Z &= f(G_1, G_2, ..., G_T) \\ &= \text{RNN}(\text{GNN}(G_1), \text{GNN}(G_2), ..., \text{GNN}(G_T) \\ &= [h_1, h_2, ..., h_T] \end{aligned}$$

Figure 3, shows the encoding architecture of Euler model.

4.3 Decoder

Decoding function is defined by:

$$\begin{aligned} g(Z_t) &= Pr(A_{t+n} = 1 | Z_t) \\ &= Z_t Z_t^T \end{aligned}$$

4.4 Loss Function

$$\begin{aligned} L_t &= -\log(Pr(A_{t+n}|Z_t) \\ &\approx \frac{-1}{|P_{t+n}|} \Sigma_{p \in P_{t+n}} \log(p) + \frac{-1}{|N_{t+n}|} \Sigma_{n \in N_{t+n}} \log(n) \end{aligned}$$

P_{t+n} refers to positive edges at snapshot $t+n$ (existing edges), and N_{t+n} refers to negative edges sampling at snapshot $t+n$ (sample of non existing edges).

4.5 Predictive tasks

In Euler model, we are interested in two different predictive tasks: link prediction and link detection. The difference occurs in the interpretation of the decoding function. With link prediction, the decoding function at timestamp t will provide $Pr(A_{t+n}|Z_t)$ with $n > 0$, while with link detection $n = 0$.

5 Addressing the flexibility and robustness of Euler model

In this section, we are going to address two problematic Cold Start and sensitivity to edges addition.

5.1 Cold Start

The goal is to improve prediction for new users. The new users are associated with nodes that have not been seen while training the model. To achieve such results,

we propose to use a node2vec model, this model is going to learn topological embeddings, and help produce richer embeddings for new nodes and thus improve the overall performance.

Given that no node features are considered (only a one hot encoding is considered as input), the prediction for new nodes might be more challenging. For link prediction, improvements are even more challenging. In fact, the unseen nodes are part of a snapshot that have never been seen while training. However, for link detection, 80% of the snapshot has been used in training (see section **??** for more details on experimental setup). For this reason Node2Vec could significantly improve the overall performance of the Euler model with link detection tasks.

We have tested two configurations. The first one consists in replacing the embeddings of new nodes encoded by GCN by Node2vec embeddings. The second approach, consists in concatenating Node2vec embeddings with GCN embedding. In section 6, we can find an experimental analysis of these approaches.

5.2 Euler sensitivity to edges insertion

The goal of this part, is to determine if the predictions given by the Euler model are impacted by abnormal edges in the learning process. In fact, for real life problems RS are encountering many connections in the historical behavior of a user that are not real interactions. This is due to some automated interactions used by third party softwares or even bots. This could introduce some biases into our model and impact its performance. To test the behavior of Euler model to these abnormal interactions, we are going to simulate these interaction by sampling some proportions of negative edges that are injected in each snapshot during the learning process. In Section 6, we are going to perform some experiments by introducing random proportion of negative edges and observe the performance of Euler model.

6 Experiments

6.1 Experimental setup

In this section we are going to evaluate Euler model in two predictive tasks: link prediction and link detection. To evaluation the model three snapshots at the end of each dataset are saved to test. The remaining snapshots are used for training and validation (80% of edges are used for training and 20% remaining for validation). The loss function is computed using the positive and negative negative sampled edges. The ratio of negative sampled edges to positive edges is 1. We are using two metrics to evaluate the model : Average Precision (AP) and Area Under the Curve (AUC). The model is tested on three datasets Enrone10 [7], DLPB [8], and FB [9]. Each experiment (training and testing) is performed 10 times.

The architecture and the parameters used are given in this section as well. We are using two GCN layers of hidden size 16 and 32 respectively. Node2vec hidden size is also 32. The hidden state of the RNN model is 32, except when we concatenate Node2Vec et GCN embedding the hidden states of RNN become 64. The Node2Vec uses 25 random walks per node, with a length of 50 for each walk.

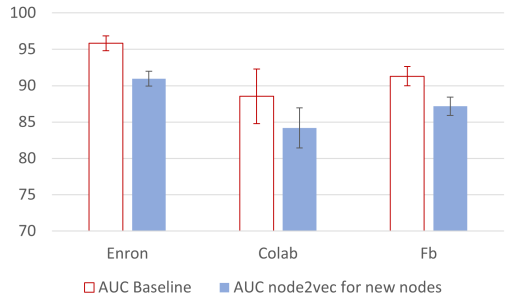
6.2 Results

In the first tests, we are comparing vanilla Euler with the results presented in [3]. In the next experiments we are evaluating two approaches used to address cold start : use node2vec as topological embedding for new nodes, and concatenate node2vec embeddings with GCN embeddings. Finally, we are going to evaluate the behavior of the model for different proportion of arbitrary edges insertion: 1%, 5%, and 10%, we decided to select proportion of edges that are not that aggressive as used in [5].

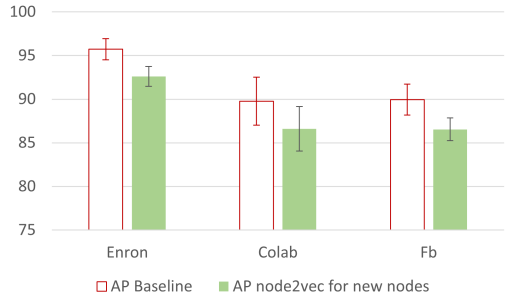
When compared to the original paper our experiment of Euler model are overall slightly lower. This hold for all dataset and both metrics. Given these inexplicable different in result, we are considering our test results are as baselines for the remaining experiments.

In addressing cold start, we have two configurations. In the first one (replacing new nodes embeddings with Node2Vec), the results decreased compared to our baseline as we can see in figures 4 and 5.

For edge detection, the AUC score dropped in average by 4,44% and AP score by 3,23%. For link prediction the drop in performance are less noticeable. The AUC score has dropped by 1,86% and the AP score by 1,1%. In the second configuration (concatenate Node2vec embedding with GCN's), there is some improvement for DBLP and FB dataset particularly for link detection as we can observe in figure 6 and 7. For Enron10 dataset, there a slight decrease in performance in both link prediction and link detection (-1,66% in average). However, for DBLP and FB datasets, we can notice some gain in link detection for both AP and AUC scores with an average increase of 2,65% for DBLP and 0,9% for FB. For link detection, the AUC scores for DBLP fluctuates around the baseline and do not have significant gain in performance. However 0.93% of average increase in AP score is noticed. It is about the same for FB dataset (no improvement for AUC and 1,47% increase in AP).

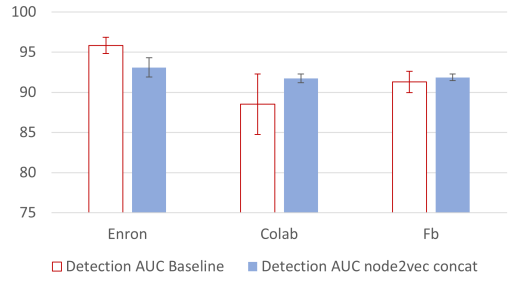


(a) AUC score for link detection

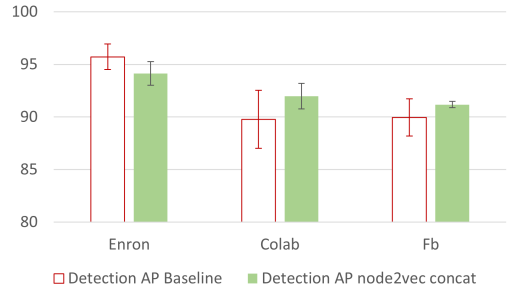


(b) AP score for link detection

Figure 4: New nodes embeddings replaced with Node2Vec for link detection

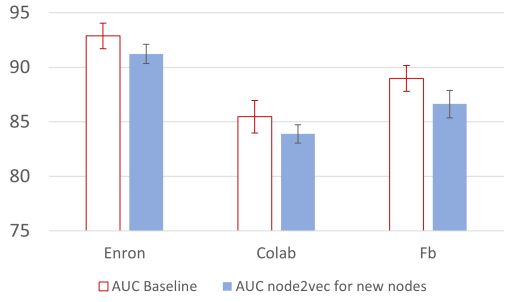


(a) AUC score for link detection

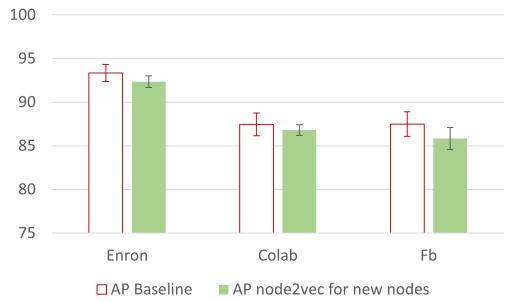


(b) AP score for link detection

Figure 6: Concatenate Node2Vec embeddings with GCN embeddings for link detection

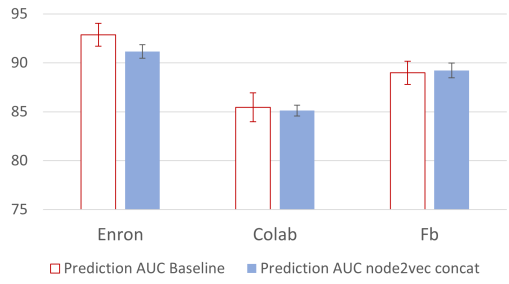


(a) AUC score for link prediction

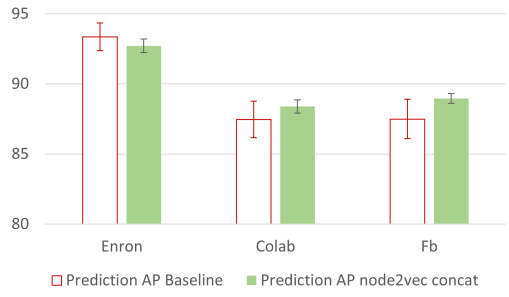


(b) AP score for link prediction

Figure 5: New nodes embeddings replaced with Node2Vec for link prediction



(a) AUC score for link prediction

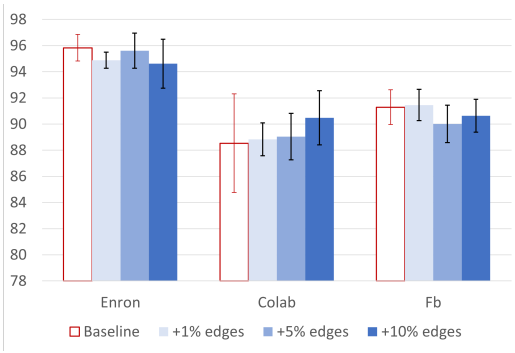


(b) AP score for link prediction

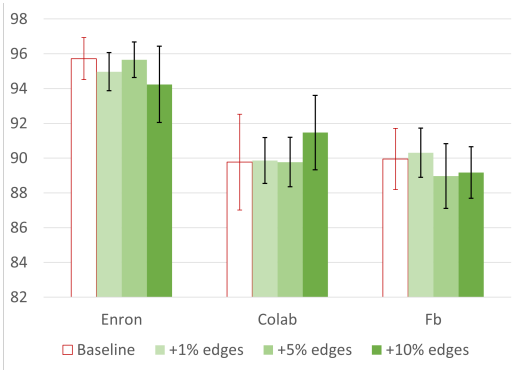
Figure 7: Concatenate Node2Vec embeddings with GCN embeddings for link prediction

This performance loss for Enron10 and performance gain for DBPL and FB, can be explained by the proportion of new nodes. In fact, Enron10, DBPL and FB gave respectively an average of 14%, 18% and 20% new nodes by test snapshots. These higher proportion of new node for both DBPL and FB dataset could partially explain the bump in performance.

For the last experiment, we have added random edges at different proportions : 1%, 5% and 10%. By increasing the proportion of edges, we can notice a drop in performance for both Enron10 and FB, as well as, an increase in the variance for FB dataset as shown in figure 8 and 9. For DBLP dataset the model has



(a) AUC score for link detection

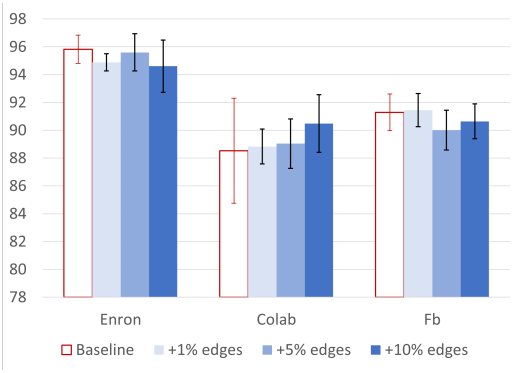


(b) AP score for link detection

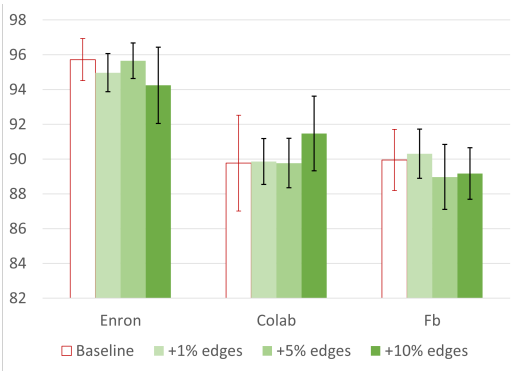
Figure 8: Euler performance with different proportion of inserted edges for link detection task

opposite behavior. While increasing the proportion of inserted edges the performance increases as well. This could be due to the fact that vanilla Euler was overfitting on the training dataset DBLP. In fact, on DBLP the difference between training score and test score is the highest among the tested dataset. Adding some noise on the training data helped decreasing the overfitting and improving the performance on test dataset.

From these experiments, we can conclude two things. First, using Node2Vec in combination with other encoder (GCN in our case) can help improve the



(a) AUC score for link prediction



(b) AP score for link prediction

Figure 9: Euler performance with different proportion of inserted edges for link prediction task

overall performance of link prediction models for certain datasets. Second, the presence of abnormal links in training datasets can negatively impact the performance of link prediction models.

7 Conclusion

During this project, we have read many articles about link prediction models for dynamic graphs in both continuous and discrete time configurations. This help us understand the principle of recommender systems using dynamic graphs, it also helped us deepening our knowledge on transductive learning through link detection. We also had the chance to experiment and explore different approaches to improve existing models. In definitive, we could improve the performance of a stat of the art model for link prediction on two datasets. We also came to the conclusion that link prediction models can be negatively affected by some bias introduced by bots and automatic requests on a network. For future work it could be interesting to perform such tests on other link prediction models and generalize these the obtained results. Random walk embedding of unseen nodes and data augmentation by adding noise are undeniably two possible future research directions.

References

[1] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. E-lstm-d: A deep learning framework for dynamic network link prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(6):3699–3712, 2019.

[2] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[3] Isaiah J King and H Howie Huang. Euler: Detecting network lateral movement via scalable temporal link prediction. *ACM Transactions on Privacy and Security*, 2023.

[4] 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 & data mining*, pages 1269–1278, 2019.

[5] Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[6] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[7] Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11:229–247, 2005.

[8] Mahmudur Rahman and Mohammad Al Hasan. Link prediction in dynamic networks using graphlet. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 394–409. Springer, 2016.

[9] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42, 2009.

[10] Leshanshui Yang, Sébastien Adam, and Clément Chatelain. Dynamic graph representation learning with neural networks: A survey. *arXiv preprint arXiv:2304.05729*, 2023.