Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Lightweight Memory Networks for Link Prediction

Generalization of the LiMNet Architecture

TITOUAN MAZIER

Lightweight Memory Networks for Link Prediction

Generalization of the LiMNet Architecture

TITOUAN MAZIER

Abstract

Lorem ipsum dolor.

Keywords

Sammanfattning

Lorem ipsum dolor.

Nyckelord

Graph Representation Learning, Temporal Interaction Networks, Link Prediction, Data Mining, Machine Learning, Recommendations

Acknowledgments

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Lorem.

Lorem ipsum dolor.

Stockholm, April 2025
Titouan Mazier

Contents

Acronyms and Abbreviations

CTDG – **Continuous-Time Dynamic Graph**: Dynamic graph were each edge is associated with a timestamp. 6

GNN – **Graph Neural Networks**: Specialized neural networks that are designed for tasks whose inputs are graphs. 5

GRL – **Graph Representation Learning**: The task of computing high level representation of graphs, subgraphs, nodes or edges. Used as inputs for Machine Learning algorithms. 4, 5

NLP – **Natural Language Processing**: Field of Machine Learning aiming at automatically interpreting spoken and written languages. 5

RNN – **Recurrent Neural Network**: A neural network whose output depends on both the inputs and on the state of an internal memory that is updated with each input. 7

Temporal Interaction Network: Dynamic network were each interaction constitute a punctual edge, i.e. an edge with no duration. 6

Chapter 1
Introduction

Lorem ipsum dolor sit amet.

1.1. Problem

Lorem ipsum.

1.2. Purpose

epoch Lorem ipsum dolor sit amet.

1.3. Goals

Lorem.

1.4. Structure of the Thesis

Lorem ipsum dolor sit amet.

Chapter 2
Background

This chapter provide the background for the project. In Section 2.1 we provide an overview of link prediction and the classical solutions for the problem, in Section 2.2 we further develop the concept of graph embeddings and some common methods to create them. Then in Section 2.3 we discuss the addition of a time dimension in graph-shaped data and the solution to exploit it, followed by a presentation of cross-RNN architectures in Section 2.4. Finally we present the model of interest for this work in Section 2.5 and why we believe that it is a relevant addition to the task of link prediction.

2.1. Link Prediction

The rapid expansion of digital content has resulted in an overwhelming abundance of information, posing significant challenges for users seeking relevant and meaningful material. A consequence to this information overload is the growing importance of search engines and recommendation systems. Among the many techniques that have emerged to tackle this challenges, one of the most used is content personalization. Instead of filtering the information in the same way for everyone, the systems will use the user's context: their search history, demographics, pasts interactions with the system, etc. to filter the information to display. For example, a system can identify the user's location through localization or search history and filter out any local information that does not match her current localization.

Content personalization can commonly be represented as a link-prediction problem in a user-item graph. In such a graph, each user and each item is associated to a node. An item can be any kind of information the user is interested in, such as web pages, music tracks, items in an e-commerce catalog, … For each interaction a user will have with the system, it will register as an edge in the graph, the goal of the personalization system is to find which item is the most relevant for a given user, which is the same as predicting which interaction should be added next in the graph.

A classical approach to that problem is to measure how close each item is to the user in the graph. Research in graph theory has provided us with a range of different ways to compute closeness between two nodes, such as measuring the shortest path connecting them, how many neighbors they share or how exclusive their common neighbors are.

The main issue with this approach is that it can only leverage structural data from the underlying graph while most of the applications provide rich context features for the users and the items. Features like such are typical inputs for Machine Learning systems, that inspired Lichtenwalter et al. to approach link prediction as a supervised problem[1]. Given a user and an item, the task is now to evaluate how likely it is that an edge will form between them in the future, based on the knowledge of the user features, the item features and a range of similarity measures computed on the past interactions graph. This typical machine learning setup leads to switch from a straightforward prediction setting to a feature engineering approach when it comes to graph-based data. Instead of looking for the desired property in the structure of the graph, this approach will try to summarize the structure into rich representation compatible with machine learning algorithm. The goal is not to proxy the desired property, but to create a rich representation of the graph data that can be used by a machine learning algorithm.

All of the previously mentioned methods presents one main drawback: each time a user want to be predicted an item, the score for each item regarding that user must be evaluated. This limitation makes it hard to scale the solution to large pools of items. A solution to limit the number of comparisons is to select a subset of potential items with a heuristic approach and only score these items. While this work well in practice, a new approach has emerged to get rid of the first heuristic selection. The key idea is to compute high-dimensional representations of the users and items separately and use simple proximity functions on these embeddings as a scoring function. This spatial representation allow to reduce the problem of finding the best item to a problem of nearest neighbor for which scalable solutions have been found.

2.2. Graph Representation Learning

The task of learning high level representation from graph data is called Graph Representation Learning, abbreviated as GRL . GRL is a general subject in data mining, it can be used to classify graph structures such as protein graphs, to capture information from a subgraph for example by

creating subgraph representation from a knowledge graph to feed into a Large Language Model. More commonly, GRL tries to create embeddings for nodes in a graph. These embeddings must capture information about the node's features but also about the context of the node in the graph, which is typically defined as the neighboring nodes and their respective features and context.

Two main approaches have emerged to create these embeddings, the first one is inspired by Natural Language Processing (NLP) and the second one from Computer vision. The first approach creates random walks along the graph and assimilate each node to a token, and each random walk to a sentence or a sample of text. This way, any methods that produce word embeddings can also be used to create nodes embeddings. This method can even be used to create embeddings for paths or subgraphs.

The second approach is inspired by convolutional networks used in computer vision. The idea is that convolutional networks approach images as graphs of pixels and successively apply transformation to these pixels based on their direct neighborhood. Such operations can be applied similarly to general graphs, the main challenge being to accept neighborhood of varying size. These architectures are called Graph Neural Networks (GNN) and have proven very effective for GRL and link-prediction[2].

2.3. Dynamic Graphs

Most of the information we get from networks is dynamic, especially for user-item interaction networks where each interaction is usually happening a a given time. Yet, when dealing with relational data, this temporal components is often disregarded to limit the complexity of the problem, or simplified as a mere feature. However temporal data constitute a unique kind of information, allowing to exploit causality relationship between the different interactions. Causality is the concept that causes will have consequences in the future. It is especially relevant for the study of interaction networks where each interaction can be the cause for a new state of each of the interacting nodes. This concept becomes critical when studying any phenomenon that can spread through the network, such as disease, information, or trend. While this concept is very intuitive for us, it is not the case for common GRL techniques described in Section 2.2 that let the information spread along the graph regardless of the order in which they are created.

In their review of dynamic network[3], Zheng et al. explain two ways temporal information are commonly included into graph data. The first one consist of considering a series of snapshot of the graph at successive time-stamps. The second one is called Continuous-Time Dynamic Graph (CTDG) . In a CTDG every edition to the graph is recorded as an event, associated with a timestamp. A typical event in a CTDG is an edge addition or deletion. For this work, the focus is on Temporal Interaction Networks , that is CTDG with all events being punctual interactions. These networks have the benefit to represent reality of a lot of system in a completely faithful way. However, the structure of the graph is blurry as each interaction correspond to a point-in-time edge that is removed immediately after happening. Because of this, we tend to approach such graphs as a stream of interactions rather than a structured network.

A popular approach to leverage temporal data when creating nodes embed-dings is to maintain a memory of the embeddings and update them as interactions are read. One of the building block for this approach is DeepCo-evolve[4] that uses a cross-RNN to update the representation of the users and items, followed by an intensity function to predict the best match at every given time t. Cross-RNN are detailed further in Section 2.4. Following DeepCoevolve, other cross-RNN models have been proposed with notables performance upgrade.

JODIE[5] is a model that builds upon DeepCoevolve by adding a static em-bedding component to the representation, using the Cross-RNN part to track the users and items trajectories, along with using a neural network layer to perform the projection operation carried over by the intensity function in DeepCoevolve.

DeePRed[6] is an other approach building on top of DeepCoevolve, this time with the aim to accelerate and simplify the training by getting rid of the recurrence in the cross-RNN mechanism. To achieve this, the dynamic embeddings, that create the recurrence, are replaced with static trained embeddings. The lack of long term information passing, is compensated by the use of a sliding context window coupled with an attention mechanism to best identify the meaningful interactions.

2.4. Cross-RNN

The key mechanism for all the aforementioned models is called cross-RNN where RNN stands for Recurrent Neural Network. A RNN is a neural network with the specificity of processing sequential data, passing an internal memory embeddings between each step of the sequence of inputs. Formally, a RNN layer is defined as

$$o(i_t) = f(i_t, h_{t-1})$$
$$h_t = g(i_t, h_{t-1})$$

Where t stands for the time step of the input i_t (i.e. it's index in the sequence). $o(i_t)$ marks the output of the layer and h_t represent the memory of the layer after receiving the input i_t. The functions f and g can vary depending on the nature of the RNN but they will rely on weights, tuned during the model training.

A Cross-RNN layer is slightly different. Instead of keeping track of a single memory embedding h_t, it maintain a memory for all nodes in the graph $H_t = (h_t^u)_{u \in \mathbb{U}} \cup (h_t^i)_{i \in \mathbb{I}}$. For each interaction $(u, i, t, \boldsymbol{f})$ the memory is updated as follow:

$$h_t^u = f^u(h_{t-1}^u, h_{t-1}^i, t, \boldsymbol{f})$$
$$h_t^i = f^i(h_{t-1}^i, h_{t-1}^u, t, \boldsymbol{f})$$

And for all other users and nodes the memory is carried over.

$$h_t^v = h_{t-1}^v \qquad \forall v \in \mathbb{U} \setminus \{u\} \cup \mathbb{I} \setminus \{i\}$$

2.5. LiMNet

Chapter 3
Method

References

[1] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '10. Washington, DC, USA: Association for Computing Machinery, 2010, pp. 243–252. doi: 10.1145/1835804.1835837.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021, doi: 10.1109/TNNLS.2020.2978386.

[3] Y. Zheng, L. Yi, and Z. Wei, "A survey of dynamic graph neural networks," *Frontiers of Computer Science*, vol. 19, no. 6, p. 196323, Dec. 2024, doi: 10.1007/s11704-024-3853-2.

[4] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep Coevolutionary Network: Embedding User and Item Features for Recommendation," no. arXiv:1609.03675, arXiv, Feb. 2017. doi: 10.48550/arXiv.1609.03675.

[5] S. Kumar, X. Zhang, and J. Leskovec, "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1269–1278. doi: 10.1145/3292500.3330895.

[6] Z. Kefato, S. Girdzijauskas, N. Sheikh, and A. Montresor, "Dynamic Embeddings for Interaction Prediction," in *Proceedings of the Web Conference 2021*, in WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 1609–1618. doi: 10.1145/3442381.3450020.