NHP: Neural Hypergraph Link Prediction

Anonymous

Abstract

Link prediction in simple graphs is a fundamental problem in which new links between nodes are predicted based on the observed structure of the graph. However, in many real-world applications, there is a need to model relationships among nodes which go beyond pairwise associations. For example, in a chemical reaction, relationship among the reactants and products is inherently higher-order. Additionally, there is need to represent the direction from reactants to products. Hypergraphs provide a natural way to represent such complex higher-order relationships. Graph Convolutional Networks (GCN) have recently emerged as a powerful deep learning-based approach for link prediction over *simple* graphs. However, their suitability for link prediction in *hypergraphs* is unexplored – we fill this gap in this paper and propose Neural Hyperlink Predictor (NHP). NHP adapts GCNs for link prediction in hypergraphs. We propose two variants of NHP – NHP-U and NHP-D – for link prediction over undirected and directed hypergraphs, respectively. To the best of our knowledge, NHP-D is the first ever method for link prediction over directed hypergraphs and NHP is also the first method for inductive link prediction in hypergraphs (can handle unseen links at test time). Through extensive experiments on multiple real-world datasets, we show NHP's effectiveness. We have made the code available to foster reproducible research.

Introduction

The problem of link prediction in graphs has numerous applications (Lu and Zhou 2011) in the fields of social network analysis (Liben-Nowell and Kleinberg 2003), knowledge bases (Nickel et al. 2016), bioinformatics (Liu et al. 2017) to name a few. However, in many real-world problems relationships go beyond pairwise associations. For example, in chemical reactions the relationship representing a group of chemical compounds that can react is inherently higher-order. Similarly, co-authorship relationships in an academic citation network are higher-order etc. Hypergraphs provide a natural way to model such higher-order complex relations. Hyperlink prediction is the problem of predicting such missing higher-order relationships in a hypergraph.

Besides the higher-order relationships, modelling the direction information between these relationships is also useful in many practical applications. For example, in the chemical reactions data, in addition to predicting groups of chemical compounds which form reactants or products, it is also important to predict the direction between reactants and products, i.e., a group of reactants react to give a group of products. Directed hypergraphs (Gallo et al. 1993) provide a way to model direction information in hypergraphs. Similar to undirected hypergraphs, predicting missing hyperlinks in a directed hypergraph is also useful in practical settings. Figure 1 illustrates the difference between modelling chemical reactions data using undirected and directed hypergraphs. Most of the previous work on hyperlink prediction (Zhou, Huang, and Schölkopf 2006; Zhang et al. 2018) focuses only on undirected hypergraphs. Moreover, they cannot handle unseen hyperlinks at test time. In this work, we focus both on undirected and directed hypergraphs.

Recently, Graph Convolutional Networks (GCNs) have emerged as a powerful tool for representation learning on graphs (Kipf and Welling 2017). GCNs have also been successfully applied for link prediction on simple graphs (Zhang and Chen 2018). Inspired by the success of GCNs for link prediction in graphs and deep learning in general (Wang, Shi, and Yeung 2017), we propose a GCN-based framework for hyperlink prediction for both undirected and directed hypergraphs. We make the following contributions:

- We propose Neural Hyperlink Predictor (NHP), a Graph Convolutional Network (GCN)-based framework, for the problem of hyperlink prediction. In contrast to previous methods, NHP can effectively handle *unseen hyperlinks at test time*, i.e., inductive link prediction in hypergraphs.
- We harness the proposed NHP for hyperlink prediction in directed hypergraphs. To the best of our knowledge, this reports the first ever attempt at the problem of *link prediction in directed hypergraphs*.
- Through extensive experiments on multiple real-world datasets, we show the effectiveness of NHP for link prediction in both undirected and directed hypergraphs.

Related work

Graph representation learning: The key advancements in learning effective node representations in graphs include matrix factorisation methods, random-walk algorithms, and deep learning on graphs (Hamilton, Ying, and Leskovec 2017b). Our work is based on deep learning on graphs.

Geometric deep learning (Bronstein et al. 2017) is an umbrella phrase for emerging techniques attempting to generalise (structured) neural network models to non-Euclidean domains such as graphs and manifolds. Graph convolutional network (GCN) (Kipf and Welling 2017) defines the convolution using a simple linear function of the graph Laplacian and is effective on graph-based semi-supervised learning (GSSL). GCNs and their extensions are the current state-of-the art for GSSL (Veličković et al. 2018; Vashishth et al. 2019; Ding, Tang, and Zhang 2018; **?**) and graph-based unsupervised learning (Hamilton, Ying, and Leskovec 2017a; Veličković et al. 2019). The reader is referred to a comprehensive literature review (Bronstein et al. 2017) and extensive surveys (Hamilton, Ying, and Leskovec 2017b; Battaglia et al. 2018; Zhang, Cui, and Zhu 2018; Wu et al.

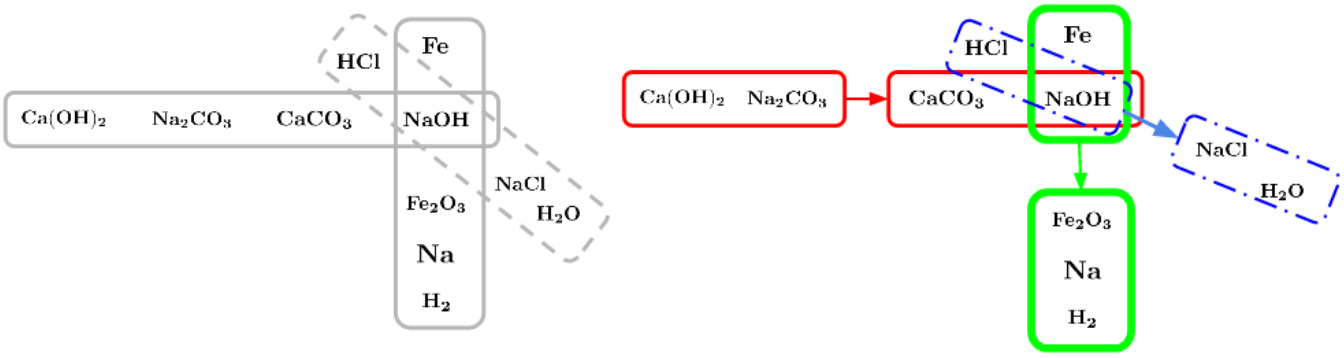

Figure 1: Difference between modelling chemical reactions using undirected and directed hypergraphs. Vertices represent chemical substances. To the left is the undirected hypergraph, in which both reactants and products are present in the same hyperlink. Whereas in the directed hypergraph (right), for a given reaction, the reactants are connected by one hyperlink and products are connected by another hyperlink and both hyperlinks are connected by a direction. Please see Introduction for more details.

2019; Sun et al. 2018; Zhou et al. 2018) on this topic. Hypergraph neural networks approximate the input hypergraph by its clique expansion and is a GCN-based method for hypergraphs (Feng et al. 2019).

Link Prediction on hypergraphs: Machine learning on hypergraphs was introduced in a seminal work (Zhou, Huang, and Schölkopf 2006) that generalised the powerful methodology of spectral clustering to hypergraphs. Link prediction on hypergraph (hyperlink prediction) has been especially popular for social networks to predict higher-order links such as *a user releases a tweet containing a hashtag* (Li et al. 2013) and to predict metadata information such as tags, groups, labels, users for entities (images from Flickr) (Arya and Worring 2018). Techniques for hyperlink prediction on social networks include ranking for link proximity information (Li et al. 2013) and matrix completion on the incidence matrix of the hypergraph (Arya and Worring 2018).

Coordinated matrix minimisation (CMM) predicts hyperlinks in the adjacency space with non-negative matrix factorisation and least square matching performed alternately in the vertex adjacency (Zhang et al. 2018). CMM uses expectation maximisation algorithm for optimisation for hyperlink prediction tasks such as predicting missing reactions of organisms' metabolic networks. For a hypergraph, it has been shown that an n-tuple-wise similarity function cannot be linear (Tu et al. 2018; **?**). Recent research has extended the notion of graph Laplacian to hypergraphs through diffusion processes for undirected (Louis 2015; Chan et al. 2018) and directed (Chan et al. 2017) hypergraphs.

Our Setup

In this section, we discuss the problem setting of hyperlink prediction in undirected and directed hypergraphs.

Undirected hyperlink prediction

An undirected hypergraph is an ordered pair $H = (V, E)$ where $V = \{v_1, \cdots, v_n\}$ is a set of n vertices and $E = \{e_1, \cdots, e_m\} \subseteq 2^V$ is a set of m hyperlinks. The problem of hyperlink prediction in the incomplete undirected hypergraph H involves predicting missing hyperlinks from $\bar{E} = 2^V - E$ based on the current set of observed hyperlinks E. Clearly, the number of vertices in any given hyperlink $e \in E$ can be any integer between 1 and 2^n. This variable cardinality of a hyperlink makes traditional link prediction methods (on simple graphs) infeasible because they are based on exactly two input features (those of the two nodes potentially forming a link).

Fortunately, in practical cases, there is no need to consider all the hyperlinks in $\bar{E}$ as most of them can be easily filtered out (Zhang et al. 2018). For example, for the task of finding missing metabolic reactions, we can restrict hyperlink prediction to all feasible reactions because the infeasible reactions seldom have biological meanings. The number of restricted hyperlinks in such practical cases is not exponential and hence hyperlink prediction on the restricted set of hyperlinks becomes a feasible problem.

Formally, a hyperlink prediction problem (Zhang et al. 2018) is a tuple $(H, \mathcal{E})$, where $H = (V, E)$ is a given incomplete hypergraph and $\mathcal{E}$ is a set of (restricted) candidate hyperlinks with $E \subseteq \mathcal{E}$. The problem is to find the most likely hyperlinks missing in H from the set of hyperlinks $\mathcal{E} - E$. The state-of-the art method for the problem is the coordinated matrix minimisation (CMM) algorithm (Zhang et al. 2018) and uses the expectation-maximisation technique to predict hyperlinks. CMM assumes the presence of all candidate hyperlinks during training and cannot handle unseen hyperlinks at test time.

Our method viz., NHP, on the other hand, does not need $\mathcal{E}$ and learns a function on the hyperlinks and hence can handle unseen hyperlinks at test time.

Directed hyperlink prediction

A directed hypergraph (Gallo et al. 1993) is an ordered pair $H = (V, E)$ where $V = \{v_1, \cdots, v_n\}$ is a set of n vertices and

$$E = \{(t_1, h_1), \cdots, (t_m, h_m)\} \subseteq 2^V \times 2^V$$

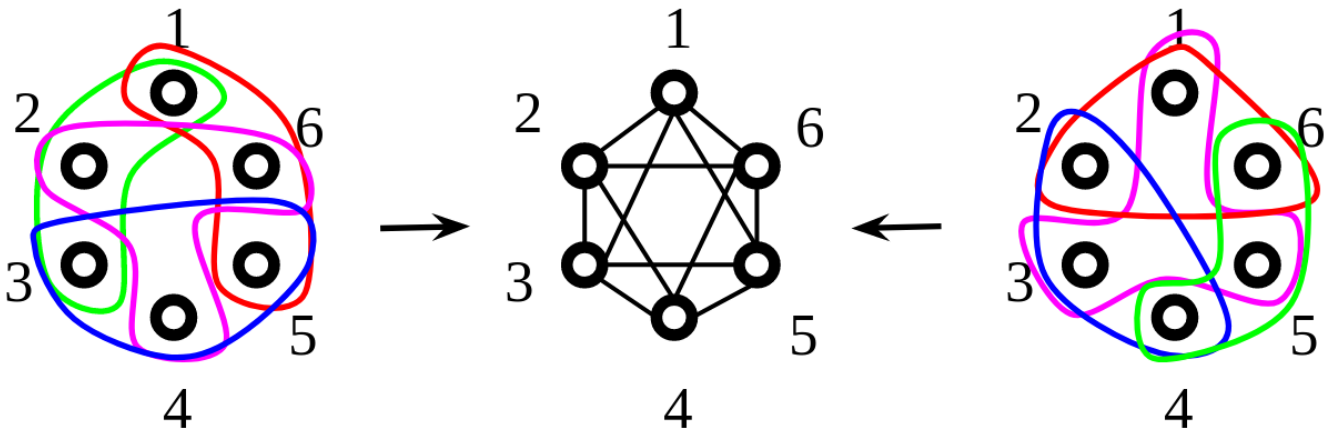

Figure 2: (best seen in colour) An example of two different hypergraphs resulting in the same weighted clique expansion. Both hypergraphs have 6 vertices and 4 hyperedges each. The hypergraph on the left side consists of $\{1,2,3\},\{1,5,6\},\{3,4,5\},\{2,4,6\}$ as its hyperedges. The hypergraph on the right contains $\{1,2,6\},\{1,3,5\},\{2,3,4\},\{4,5,6\}$ as its hyperedges. The two hypergraphs have the same connections in their clique expansions (as shown in the middle). Moreover, the (normalised) weights on the edges of the clique expansions are also the same ($\frac{2}{3}$ each). Please see Section NHP: Neural Hyperlink Predictor for more details.

is a set of m directed hyperlinks. Each $e \in E$ is denoted by (t, h) where $t \subseteq V$ is the *tail* and $h \subseteq V$ is the *head* with $t \neq \Phi$, $h \neq \Phi$. As shown in Figure 1, chemical reactions can be modelled by directed hyperlinks with chemical substances forming the set V. A directed simple link is the special case when $|t| = |h| = 1$. Given an incomplete directed hypergraph $H = (V, E)$, the problem of directed hyperlink prediction is to predict the missing hyperlinks in H.

NHP: Neural Hyperlink Predictor

In this section, we explain the proposed framework NHP. NHP-U refers to the setting of undirected hyperlink prediction and NHP-D refers to the directed setting.

NHP-U

A schematic view of NHP-U is shown in Figure 3. It consists of three components, viz. network embedding, hyperlink-aware embedding, and a tuple-wise interaction function.

Network embedding initialisation: We convert the input (incomplete) hypergraph into its clique expansion (Zhou, Huang, and Schölkopf 2006; Agarwal, Branson, and Belongie 2006). The clique expansion of a hypergraph introduces a (weighted / normalised) clique for each hyperlink of the hypergraph. Once the clique expansion is obtained, popular unsupervised embedding methods such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015), and node2vec (Grover and Leskovec 2016) can be used to obtain the initial embeddings of the vertices. Alternatively, one may use trainable deep models such as graph convolutional networks (GCNs) (Kipf and Welling 2017), or graph attention networks (Veličković et al. 2018) to obtain the initial embeddings.

Hyperlink-aware embedding (GCN layer): The clique expansion of a hypergraph introduces pairwise connections between every vertex pair in each hyperlink of the hypergraph. Because of the pairwise connections, the embedding initialisations obtained from the previous component do not consider each hyperlink as a unit (in which vertex connections go beyond pairwise). As a simple example two hypergraphs $H_1 = (V, E_1)$ and $H_2 = (V, E_2)$ with $V = \{1,2,3,4,5,6\}$, $E_1 = \{1,2,3\},\{1,5,6\},\{3,4,5\},\{2,4,6\}\}$ and $E_2 = \{1,2,6\},\{1,3,5\},\{2,3,4\},\{4,5,6\}$ result in the same weighted clique expansions and hence the network embedding initialisations would be the same.

We propose to address the above issue by refining the embedding initialisations with a GCN layer on the subgraph obtained from the clique expansion of each hyperlink and then passing the embeddings to a trainable tuple-wise interaction function. In other words, given a hyperlink e, we refine the embedding of each vertex $v \in e$ using the following GCN equation of neural message-passing (Gilmer et al. 2017):

$$h_v^{(e)} = \text{ReLU}\Big(W_{GCN} \sum_{u \in e} x_u + b_{GCN}\Big) \qquad (1)$$

where x_v is the network embedding initialisation obtained from the previous component. This is highlighted by different colours in Figure 3. We then pass the set of refined embeddings $\{h_v^{(e)} : v \in e\}$ for each hyperlink $e \in E$ to a tuple-wise interaction function described below.

Tuple-wise interaction function (INT layer): We note that the previous component, viz. hyperlink-aware embedding, still does not preserve the higher order relationships among the vertices of a given hyperlink $e \in E$. We thus propose to use a trainable interaction function, I_e, to preserve

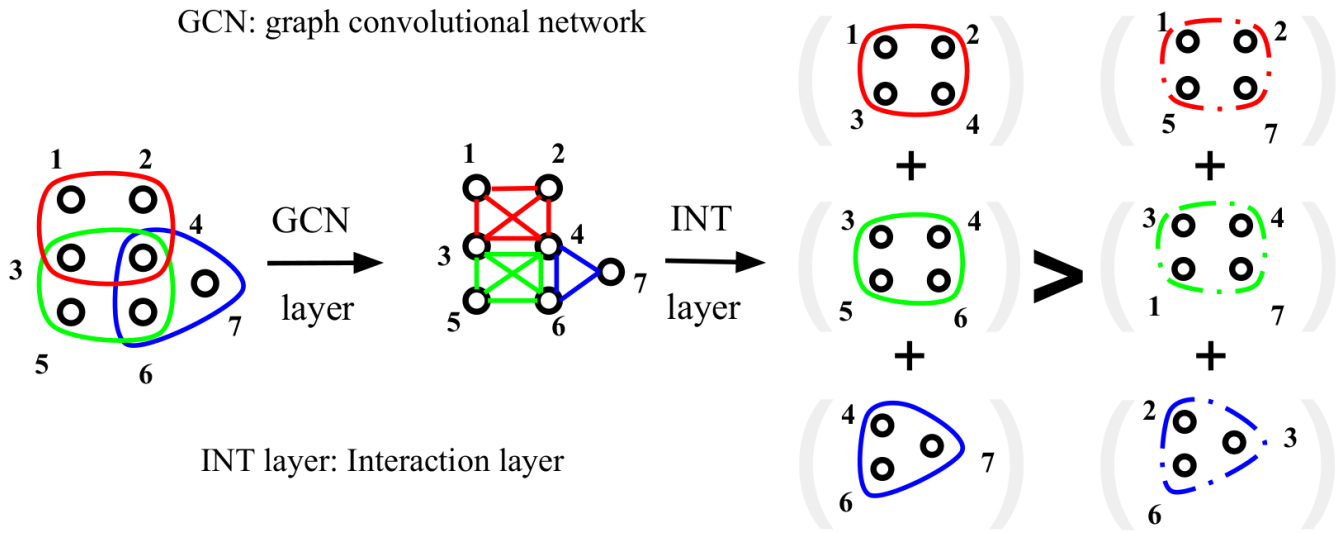

Figure 3: (best seen in colour) NHP for undirected hyperlink prediction. The input hypergraph is converted to its clique expansion to get the initial embeddings on the vertices of the hypergraph. The initialisations are then fed to a GCN layer that refines the embeddings on the subgraph obtained from the clique expansion of each hyperlink (highlighted by different colours). The INT layer is the interaction layer which assigns a score to each hyperlink. The interaction score on each hyperlink is desired to be higher than that for any set of vertices that does not form a hyperlink. Please see the section entitled NHP: Neural Hyperlink Predictor for more details.

the higher-order relationships in the input hypergraph. The function takes the form

$$I_e = \sigma\Big(W \cdot f\Big(\{h^{(e)}\}_{v\in e}\Big) + b\Big) \tag{2}$$

where W is a parameter of dimension $1 \times d$ and σ is the sigmoid function. Intuitively, the interaction score, I_e, for hyperlink e, ideally, needs to be higher than that for any set of vertices that does not form a hyperlink in the hypergraph.

The value I_e is high if the learned W is well-aligned to the function f. Thus we propose to use the mean of the embeddings of a hyperlink e as the function f i.e.,

$$I_e := \sigma\Big(\frac{1}{|e|}W \cdot \sum_{v\in e} h_v^{(e)} + b\Big). \tag{3}$$

However, the function in Equation 3 does not guarantee that the vertices in a hyperlink have similar embeddings. Motivated by the similarity-based hypergraph Laplacian proposed in literature (Louis 2015; Chan et al. 2018), we also propose the following function, which ensures that the vertices in a hyperlink have similar embeddings.

$$I_e := \sigma\Big(\frac{1}{|e|}W \cdot \text{maxmin}\{h_v^{(e)}\}_{v\in e} + b\Big). \tag{4}$$

where given a set of vectors $x_1, \cdots, x_k \in \mathbb{R}^d$, the function

$$\text{maxmin}\{x_j : j \in [k]\} = (\max_{s\in[k]} x_{sl} - \min_{i\in[k]} x_{il})_{l=1,\cdots,d} \tag{5}$$

is the element-wise difference of maximum and the minimum values of the vectors. The tuple-wise interaction function is precisely what enables NHP to handle unseen hyperlinks at test time which the existing approaches (Zhang et al. 2018; Zhou, Huang, and Schölkopf 2006) cannot.

Optimisation: Hyperlinks in the input hypergraph represent known interactions among the vertices of the hyperlink. The set of unknown interactions i.e., $2^V - E$ may, in fact, contain undiscovered hyperlinks and belong to the existing ones. Following prior work (Liu et al. 2017), we rely on a ranking objective as follows:

$$\mathcal{L} = \frac{1}{|E|}\sum_{e\in E} \Lambda\Big(\frac{1}{|F|}\sum_{f\in F} I_f - I_e\Big). \tag{6}$$

where F is a set of sampled vertex sets from $2^V - E$. The notation $\Lambda(x)$ is used to denote a non-decreasing function such as the popular logistic function $\Lambda(x) = \log(1+e^x)$. The loss $\mathcal{L}$ above tries to maximise the number of hyperlink interaction scores (in E) that are higher than the average interaction score of the unknown vertex sets in F. It ranks the observed hyperlinks above the unobserved ones. The loss function in Equation 6 is more reliable than a strict binary classification objective as pointed out in the prior work (Liu et al. 2017). All parameters / weights of NHP-U i.e W_{GCN} and W are learned end-to-end using stochastic gradient descent.

Sampling method: We give the method to construct the set F. For each hyperlink $e \in E$, we create a corresponding $f \in F$ by having half of the vertices i.e., $\frac{|e|}{2}$ sampled from e and the remaining half from $V - e$. This sampling method

is motivated by the chemical reaction datasets where it is highly unlikely that half of the substances of a valid reaction (from e) and randomly sampled substances (from $V - e$) are involved in another valid reaction. To avoid any possible bias in the hyperlink sizes, we ensure that each hyperlink e has a "corresponding" vertex set f of the same size. We now extend the proposed approach for directed hypergraphs.

NHP-D

NHP-D consists of four components three of which are network embedding, hyperlink-aware embedding, and a tuple-wise interaction function. The fourth component is the direction prediction component to predict direction between two hyperlinks (e.g.: direction from reactants to products in reaction data). We follow the same procedure for the network embedding initialisation and hyperlink-aware embedding as described for NHP-U. Note that for initialisation, we treat each directed hyperlink as an undirected hyperlink (union of tail and head). For the tuple-wise interaction function, we use the difference between the element-wise maximum in tail and minimum in head hyperlinks (Zhang et al. 2017; Chan et al. 2017). In other words, for a directed hyperlink $e = (t, h)$, $I_e = \sigma(W \cdot \text{maxmin}_+\{h_v^{(e)}\}_{v \in e} + b)$ where

$$\text{maxmin}_+\{x_v\}_{v \in e} = (\max_{s \in t} x_{sl} - \min_{i \in h} x_{il})_{+, l=1,\cdots,d}$$

and $m_+ = m$ if $m > 0$ and $m_+ = 0$ if $m \leq 0$.

Direction prediction: We propose the bilinear form

$$D_{pq} = \sigma\Big(\mathbf{p}^T \ W_{BL} \ \mathbf{q} + b_{BL}\Big) \tag{7}$$

to predict direction between two hyperlinks where $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$ are the hyperlink embeddings of hyperlinks $p, q \subseteq V$ respectively, σ is the sigmoid non-linearity and $W_{BL} \in \mathbb{R}^{d \times d}$, b_{BL} are the bilinear weight and bias respectively. We use the mean of the embeddings of a hyperlink to get the embedding of the hyperlink i.e., $\mathbf{p} = \sum_{v \in p} h_v^{(p)}$. We use a similar ranking objective to predict directions i.e.

$$\mathcal{L}_{dir} = \frac{1}{|E|} \sum_{(t,h) \in E} \Lambda\left(\frac{1}{|F|} \sum_{(p,q) \in F} D_{pq} - D_{th}\right) \tag{8}$$

We use a joint optimisation strategy to get the tuple-wise interaction score I_e and the direction score D_{th} for an ordered pair $e = (t, h)$ i.e. we minimise $\mathcal{L} + \lambda \mathcal{L}_{dir}$ using back-propagation. The parameters of the model i.e. W_{GCN}, W, W_{BL} are all updated end-to-end using back-propagation.

Inference: At test time, NHP-U and NHP-D predict a hyperlink e as positive (existing) if its score is higher than the average score of the unobserved links (used for training), otherwise it predicts it as negative (non-existing). A similar step is used by NHP-D for predicting directions.

Computational complexity

For the given input (incomplete) hypergraph (V, E), let

$$N_1 = \sum_{e \in E} |e| \quad \text{and} \quad N_2 = \sum_{e \in E} \frac{1}{2} \cdot |e| \cdot \Big(|e| - 1\Big)$$

where for a directed hyperedge $(t, h) \in E$, we define $|e| := |t| + |h|$.

Our negative sampling strategy (pre-processing step) takes $O(N_1)$ time. The GCN layer takes $O(N_2)$ time, the INT layer and additionally the bilinear layer for NHP-D take $O(N_1)$ time each. Once all the hyperedge scores are obtained, computing the loss takes $O(|E|)$ time. Hence the computation complexity of NHP is $O(N_1 + N_2 + |E|)$.

Assuming sparse real-world hypergraphs, i.e., $|E| \in O(|V|)$ and $|e| \in O(1)$ for each hyperedge $e \in E$, our NHP takes $O(|V|)$ time.

Datasets and motivation

We used a knowledge graph, a co-authorship network, and three chemical reactions networks (one organic and two metabolic) as datasets for our experiments. The statistics of the datasets are shown in Table 1. Note that we used chemical reaction networks as both directed and undirected hypergraphs. We used the knowledge graph as a directed hypergraph and a co-authorship network as an undirected hypergraph for a total of 4 undirected and 4 directed hypergraph experiments. The motivation and the construction of these datasets are pushed to the supplementary.

Experiments

On all the real-world datasets, we report mean AUC and Recall@k (k is the number of missing links) numbers averaged over 10 random splits of train and test set of hyperlinks. 90% hyperlinks in each dataset was used for training and the remaining 10% for testing the proposed models and all the baselines. Hyper-parameters were optimised using cross-validation and are shown in the supplementary material.

Experiments on directed hypergraphs

Since there are no proposed approaches for the problem, we compared against the following baselines:

- **node2vec:** we used node2vec embeddings with the following tuple wise interaction function:

$$I_e = \sigma\Big(\frac{1}{|e|} \sum_{u,v \in e \times e - \{(w,w): w \in e\}} x_u^T x_v\Big)$$

  where $e = t \cup h$ The above function considers all vertex pairs for each hyperedge.

- **node2vec-GCN:** we used node2vec embeddings as initialisation to a GCN and then train the GCN's parameters with the previously mentioned interaction function. Note that this baseline does not use the proposed tuple-wise interaction functions viz., mean and maxmin$_+$.

- **node2vec-mean:** we used node2vec embeddings as initialisation to a trainable tuple wise interaction function which takes the mean of the vertex embeddings to learn the weights

Table 1: Summary of the real-world hypergraph datasets used in the experiments.

Dataset	**Reverb45k**	**DBLP**	**USPTO**	**iJO1366**	**iAF1260b**
type of data	knowledge graph	co-authorship	organic reactions	metabolic reactions	metabolic reactions
type of hypergraph	directed	undirected	directed, undirected	directed, undirected	directed, undirected
number of vertices	28798	20685	16293	1805	1668
number of hyperlinks	66914	44337	11433	2583	2388

dataset $\rightarrow$	iAF1260b		iJO1366		USPTO	
model $\downarrow$	AUC	Recall@k	AUC	Recall@k	AUC	Recall@k
node2vec	0.52 ± 0.01	0.14 ± 0.05	0.52 ± 0.03	0.20 ± 0.03	0.53 ± 0.04	0.16 ± 0.02
node2vec-GCN	0.53 ± 0.01	0.17 ± 0.03	0.52 ± 0.01	0.23 ± 0.03	0.56 ± 0.03	0.18 ± 0.02
node2vec-mean	0.52 ± 0.03	0.18 ± 0.03	0.51 ± 0.03	0.22 ± 0.04	0.56 ± 0.04	0.17 ± 0.04
node2vec-maxmin$_+$	0.53 ± 0.01	0.21 ± 0.01	0.52 ± 0.01	0.24 ± 0.01	0.58 ± 0.02	0.24 ± 0.03
NHP-D-mean	0.55 ± 0.01	0.23 ± 0.05	0.54 ± 0.02	0.26 ± 0.02	0.60 ± 0.03	0.18 ± 0.03
NHP-D-maxmin$_+$	$\mathbf{0.58 \pm 0.02}$	$\mathbf{0.26 \pm 0.04}$	$\mathbf{0.56 \pm 0.01}$	$\mathbf{0.28 \pm 0.03}$	$\mathbf{0.63 \pm 0.02}$	$\mathbf{0.25 \pm 0.04}$

Table 2: Mean AUC and Recall@k values (higher is better) for link prediction in directed hypergraphs on the three chemical reaction datasets. Our proposed method achieves superior performance compared to all the baselines.

model	AUC	Recall@k
node2vec	0.57 ± 0.01	0.40 ± 0.04
node2vec-GCN	0.62 ± 0.02	0.42 ± 0.03
node2vec-mean	0.65 ± 0.02	0.44 ± 0.05
node2vec-maxmin$_+$	0.75 ± 0.01	0.63 ± 0.04
NHP-D-mean	0.72 ± 0.05	0.45 ± 0.03
NHP-D-maxmin$_+$	$\mathbf{0.81 \pm 0.04}$	$\mathbf{0.65 \pm 0.03}$

Table 3: Average AUC and Recall@k values on the Reverb45k knowledge graph.

- **node2vec-maxmin$_+$:** we used node2vec embeddings as initialisation to a trainable tuple wise interaction function which takes the element-wise difference of maximum (in tail) and minimum (in head) of the vertex embeddings to learn the weights. This baseline and the previous baseline do not have the GCN layer.

We used the direction prediction component with all the aforementioned baselines and learn the parameters W_{BL}. The results are shown in Table 2. As we can see, our proposed NHP-D is able to outperform all the baselines justifying all the components in our model. We also observe that the models that use maxmin$_+$ outperform the mean counterparts. This shows that making the vertex embeddings similar makes it more effective to learn known interactions in the hypergraph.

The results for the Reverb45k knowledge graph are shown in Table 3. At test time, we introduced 10% directed hyperlinks which were clearly not canonicalised (e.g.: tail containing {Obama, Trump}) and head containing {UK, Brazil}). The set of 10% missing hyperlinks union the above set of negative hyperlinks formed the entire test set.

Statistical test We performed a Welch t-test (Welch 1947) on our results. We compared our proposed best method NHP-D-maxmin $_+$ with the most competitive baselines. The p-values for all experiments and metrics (AUC and Recall@k) in Tables 2 and 3 were lower than 0.05 except the following two:

- Recall@k for USPTO: $p = 0.54$
- Recall@k for Reverb45k: $p = 0.22$

This demonstrates the statistical significance of our results.

Experiments on undirected hypergraphs

The state-of-the-art methods for link prediction in undirected hypergraphs are Co-ordinated matrix minimisation (CMM) (Zhang et al. 2018), and Spectral Hypergraph Clustering (SHC) (Zhou, Huang, and Schölkopf 2006). We compared NHP-U against the following baselines:

- **CMM (Zhang et al. 2018):** This baseline uses the expectation-maximisation algorithm in the adjacency space to predict hyperlinks. It is inherently a transductive algorithm i.e. it cannot handle unseen links at test time.
- **SHC (Zhou, Huang, and Schölkopf 2006):** This baseline converts the input hypergraph into its dual so that hyperlink prediction can be posed as semi-supervised transductive vertex classification in the dual. It also inherently cannot handle unseen links at test time.

  We note that the two undirected methods viz., CMM and SHC, are not embedding-based methods and hence, the direction prediction component cannot be trivially used to predict hyperlinks in *directed hypergraph experiments*.
- **node2vec:** we used node2vec embeddings with the following tuple wise interaction function:

$$I_e = \sigma\Big(\frac{1}{|e|} \sum_{u,v \in e \times e - \{(w,w): w \in e\}} x_u^T x_v\Big)$$

  The above function considers all vertex pairs for each hyperedge.
- **node2vec-GCN:** we used node2vec embeddings as initialisation to a GCN and then train the GCN's parameters with the previously mentioned interaction function. Note

data	dataset →	iAF1260b		iJO1366		USPTO	
	model ↓	AUC	Recall@k	AUC	Recall@k	AUC	Recall@k
H	node2vec	0.57 ± 0.03	0.21 ± 0.05	0.53 ± 0.04	0.23 ± 0.03	0.53 ± 0.05	0.22 ± 0.03
H	node2vec-GCN	0.59 ± 0.04	0.26 ± 0.03	0.56 ± 0.02	0.27 ± 0.02	0.56 ± 0.02	0.24 ± 0.04
H	node2vec-mean	0.58 ± 0.03	0.24 ± 0.04	0.54 ± 0.02	0.27 ± 0.03	0.58 ± 0.03	0.23 ± 0.02
H	node2vec-maxmin	0.61 ± 0.05	0.28 ± 0.03	0.60 ± 0.02	0.29 ± 0.03	0.68 ± 0.03	0.26 ± 0.01
$(H, \mathcal{E})$	SHC	$\mathbf{0.65 \pm 0.01}$	$\mathbf{0.31 \pm 0.02}$	$\mathbf{0.64 \pm 0.01}$	$\mathbf{0.33 \pm 0.02}$	0.56 ± 0.01	0.22 ± 0.01
$(H, \mathcal{E})$	CMM	$\mathbf{0.64 \pm 0.04}$	$\mathbf{0.30 \pm 0.14}$	$\mathbf{0.64 \pm 0.03}$	$\mathbf{0.35 \pm 0.10}$	0.68 ± 0.01	$\mathbf{0.37 \pm 0.01}$
H	NHP-U-mean	0.60 ± 0.04	0.28 ± 0.06	0.61 ± 0.02	0.29 ± 0.02	0.65 ± 0.02	0.20 ± 0.05
H	NHP-U-maxmin	$\mathbf{0.64 \pm 0.03}$	$\mathbf{0.31 \pm 0.03}$	$\mathbf{0.63 \pm 0.02}$	$\mathbf{0.32 \pm 0.02}$	$\mathbf{0.74 \pm 0.02}$	0.29 ± 0.02

Table 4: Mean AUC and Recall@k values (higher is better) for link prediction in undirected hypergraphs on the three chemical reaction datasets. Our proposed method achieves comparable performance compared to the state-of-the-art baselines. SHC and CMM need all the candidate hyperlinks during training and cannot handle unseen hyperlinks at test time. Our method achieves comparable performance without using the candidates and can generalise to unseen hyperlinks at test time.

that this baseline does not use the proposed tuple-wise interaction functions viz., mean and maxmin$_+$.

- **node2vec-mean:** we used node2vec embeddings as initialisation to a trainable tuple wise interaction function which takes the mean of the vertex embeddings to learn the weights
- **node2vec-maxmin:** we used node2vec embeddings as initialisation to a trainable tuple wise interaction function which takes the element-wise difference of maximum and minimum of the vertex embeddings in each hyperedge to learn the weights. This baseline and the previous baseline do not have the GCN layer.

Table 4 shows the results of the experiments on the undirected hypergraphs. Our results demonstrate strong performance across all the datasets. The state-of-the art baselines viz., Co-ordinated matrix minimisation (CMM) (Zhang et al. 2018), and Spectral Hypergraph Clustering (SHC) (Zhou, Huang, and Schölkopf 2006), require all candidate hyperlinks (for reaction datasets) to be present during training. Our proposed NHP and all the other proposed baselines, on the other hand, can handle unseen hyperlinks at test time. We particularly note that NHP is competitive with the results of SHC and CMM even if NHP does not use the set of candidate hyperlinks, $\mathcal{E}$, during training. We assume that these benefits stem from the fact that, the proposed tuple-wise interaction function is trainable from the input hypergraph. The interaction function enables us to handle unseen hyperlinks at test time.

Results on DBLP co-authorship Table 5 shows the results. In a coauthorship network, there is no notion of "candidate" set of authors (which include "negative" collaborations) who could potentially collaborate. SHC relies on a strict binary classification objective and hence requires negative links. CMM requires candidate hyperlinks for training. Hence these two baselines cannot be meaningfully used on this dataset.

NHP-U, however, uses a ranking objective and the interaction function (INT layer) is precisely what enables it to handle unseen hyperlinks at test time. Hence our method and our proposed baselines can be used on this dataset. The p-value (Welch t-test) of NHP-U-maxmin on this dataset is less than 0.01 with the most competitive baseline and hence this demonstrates the statistical significance.

model	Recall@k
node2vec	0.38 ± 0.03
node2vec-GCN	0.41 ± 0.04
node2vec-mean	0.41 ± 0.03
node2vec-maxmin	0.45 ± 0.06
SHC	**NA**
CMM	**NA**
NHP-U-mean	0.46 ± 0.04
NHP-U-maxmin	$\mathbf{0.51 \pm 0.05}$

Table 5: Average Recall@k values on dblp co-authorship network. **NA**: not applicable. SHC relies on a strict binary classification objective while CMM requires candidates.

Approximate training time comparison On the largest chemical reaction dataset, USPTO, NHP-U takes around 12 hours of training time, while SHC and CMM take around 1 day and 3 days respectively. The faster training time is due to GPU compatibility (NHP is GPU friendly while SHC and CMM are not).

Conclusion and future work

We have introduced NHP, a novel neural approach for hyperlink prediction in both undirected and directed hypergraphs. To the best of our knowledge, this is the first method for link prediction in directed hypergraphs. NHP can effectively handle unseen hyperlinks at test time.

Our NHP framework can easily use graph attention networks (Veličković et al. 2018) instead of GCNs and DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and LINE (Tang et al. 2015) instead of node2vec. A future direction is to explore multi-relational directed hypergraphs in which directions can have labels.

References

Agarwal, S.; Branson, K.; and Belongie, S. 2006. Higher order learning with graphs. In *ICML*.

Arya, D., and Worring, M. 2018. Exploiting relational information in social networks using geometric deep learning on hypergraphs. In *ICMR*.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR, arXiv:1806.01261*.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*

Chan, T. H.; Tang, Z. G.; Wu, X.; and Zhang, C. 2017. Diffusion operator and spectral analysis for directed hypergraph laplacian. *CoRR, arXiv:1711.01560*.

Chan, T.-H. H.; Louis, A.; Tang, Z. G.; and Zhang, C. 2018. Spectral properties of hypergraph laplacian and approximation algorithms. *J. ACM*.

Ding, M.; Tang, J.; and Zhang, J. 2018. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 913–922.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph neural networks. In *AAAI*.

Gallo, G.; Longo, G.; Pallottino, S.; and Nguyen, S. 1993. Directed hypergraphs and applications. *Discrete Appl. Math.*

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *KDD*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017a. Inductive representation learning on large graphs. In *NIPS*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Li, D.; Xu, Z.; Li, S.; and Sun, X. 2013. Link prediction in social networks based on hypergraph. In *WWW*.

Liben-Nowell, D., and Kleinberg, J. 2003. The link prediction problem for social networks. In *CIKM*.

Liu, Y.; Qiu, S.; Zhang, P.; Gong, P.; Wang, F.; Xue, G.; and Ye, J. 2017. Computational drug discovery with dyadic positive-unlabeled learning. In *ICDM*.

Louis, A. 2015. Hypergraph markov operators, eigenvalues, approximation algorithms. In *STOC*.

Lu, L., and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*.

Sun, L.; Wang, J.; Yu, P. S.; and Li, B. 2018. Adversarial attack and defense on graph data: A survey. *CoRR, arXiv:1812.10528*.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*.

Tu, K.; Cui, P.; Wang, X.; Wang, F.; and Zhu, W. 2018. Structural deep embedding for hyper-networks. In *AAAI*.

Vashishth, S.; Yadav, P.; Bhandari, M.; and Talukdar, P. 2019. Confidence-based graph convolutional networks for semi-supervised learning. In *AISTATS*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Lió, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep graph infomax. In *ICLR*.

Wang, H.; Shi, X.; and Yeung, D. 2017. Relational deep learning: A deep latent variable model for link prediction. In *AAAI*.

Welch, B. L. 1947. The generalization of 'student's' problem when several different population variances are involved. *Biometrika* 34(1/2):28–35.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *CoRR, arXiv:1901.00596*.

Zhang, M., and Chen, Y. 2018. Link prediction based on graph neural networks. In *NIPS*.

Zhang, C.; Hu, S.; Tang, Z. G.; and Chan, T.-H. H. 2017. Re-revisiting learning on hypergraphs: Confidence interval and subgradient method. In *ICML*.

Zhang, M.; Cui, Z.; Jiang, S.; and Chen, Y. 2018. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*.

Zhang, Z.; Cui, P.; and Zhu, W. 2018. Deep learning on graphs: A survey. *CoRR, arXiv:1812.04202*.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *CoRR, arXiv: 1812.08434*.

Zhou, D.; Huang, J.; and Schölkopf, B. 2006. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*.