NHP: Neural Hypergraph Link Prediction

Naganand Yadati
naganand@iisc.ac.in
Indian Institute of Science
Bangalore, Karnataka, India

Vikram Nitin
vikram.nitin@columbia.edu
Columbia University
New York, United States of America

Madhav Nimishakavi
madhavn@FB.com
Facebook AI
London, United Kingdom

Prateek Yadav
prateekyadav.iisc@gmail.com
LinkedIn
Bangalore, Karnataka, India

Anand Louis
anandl@iisc.ac.in
Indian Institute of Science
Bangalore, Karnataka, India

Partha Talukdar
ppt@iisc.ac.in
Indian Institute of Science
Bangalore, Karnataka, India

ABSTRACT

Link prediction in *simple graphs* is a fundamental problem in which new links between vertices are predicted based on the observed structure of the graph. However, in many real-world applications, there is need to model relationships among vertices 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 underexplored – 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. An important feature of NHP is that it can also be used for hyperlinks in which dissimilar vertices interact (e.g. acids reacting with bases). Another attractive feature of NHP is that it can be used to predict unseen hyperlinks at test time (inductive hyperlink prediction). Through extensive experiments on multiple real-world datasets, we show NHP's effectiveness. The code is available at **this link**

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Unsupervised learning*.

KEYWORDS

Link Prediction, Directed Hypergraph, Graph Neural Network, Knowledge Graph Canonicalisation

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411870

ACM Reference Format:
Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2020. NHP: Neural Hypergraph Link Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411870

1 INTRODUCTION

The problem of link prediction in graphs has numerous applications [20] in the fields of social network analysis [25], knowledge bases [29], bioinformatics [26] 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 network are high-order. Hypergraphs provide a natural way to model such relationships. Hyperlink prediction [1] is the problem of predicting missing high-order relationships in a hypergraph.

Besides higher-order relationships, modelling directions between these relationships is also useful in many practical applications. For example, in chemical reactions, in addition to predicting groups of chemical compounds which form reactants or products, it is also important to predict directions between reactants and products, i.e., a group of reactants react to give a group of products. Directed hypergraphs [12] provide a way to model directions in hypergraphs. Similar to undirected hypergraphs, predicting missing hyperlinks in directed hypergraphs is also useful in practice. Difference between undirected and directed hypergraphs is illustrated in Figure 1. Previous works on hyperlink prediction [44–46] focus only on undirected hypergraphs with small hyperlink sizes. Moreover, most of these works cannot handle unseen hyperlinks at test time. In this work, we focus both on undirected and directed hypergraphs containing hyperlinks of arbitrary sizes.

Recently, Graph Convolutional Networks (GCNs) have emerged as a powerful tool for representation learning on graphs [19]. GCNs have also been successfully applied for link prediction on simple graphs [43]. Inspired by the success of GCNs for link prediction in graphs, we propose a novel GCN-based framework for hyperlink prediction for both undirected and directed hypergraphs. We make the following contributions:

[1] We note that hyperedge can be used synonymously with hyperlink.

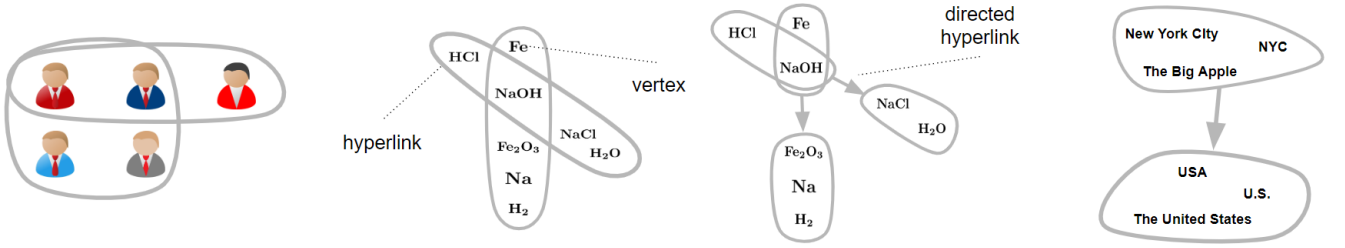

Figure 1: Real-world hypergraph datasets for hyperlink prediction. a) shows a co-authorship network with authors as vertices and collaborations as hyperlinks. b) shows a chemical reaction network with substances as vertices and reactions as hyperlinks. c) shows the directed hypergraph version of b) in which reactants and products are hyperlinks connected by a direction. d) shows a canonicalised knowledge link modelled as a directed hyperlink in which words/phrases are vertices, canonicalised words/phrases form a hyperlink and subject hyperlink (e.g. New York City) is related to an object hyperlink (e.g. U.S.) by a direction. Please See Introduction for more details.

- We propose Neural Hyperlink Predictor (NHP), a novel GCN-based framework, for hyperlink prediction. NHP uses novel scoring functions to rank existing hyperlinks higher than non-existing vertex sets. A key novelty of NHP (see Proposition 1) is that it can also be used for hyperedges in which dissimilar vertices interact (e.g. acids reacting with bases). In contrast to non-neural baselines, NHP can predict unseen hyperlinks at test time (inductive hyperlink prediction).
- 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*. Novel scoring functions of NHP can flexibly be used in existing deep hypergraph methods for link prediction in directed (and undirected) hypergraphs.
- Through extensive experiments on multiple real-world datasets, we show the effectiveness of NHP for link prediction in both undirected and directed hypergraphs. NHP's source code can be downloaded from **this link**

2 RELATED WORK

In this section, we briefly review related work in deep learning on graphs and link prediction on hypergraphs.

Learning representations on graphs: The key advancements in learning low-dimensional vertex representations in graphs include matrix factorisation-based methods, random-walk based algorithms, and deep learning on graphs [16]. Our work is based on deep learning on graphs.

Geometric deep learning [5] is an umbrella phrase for emerging techniques attempting to generalise (structured) deep neural network models to non-Euclidean domains such as graphs and manifolds. Graph convolutional network (GCN) [19] defines the convolution using a simple linear function of the graph Laplacian and is shown to be effective on semi-supervised classification on attributed graphs. GCNs and their extensions are the current state-of-the art for graph-based semi-supervised learning [33, 34] and graph-based unsupervised learning [15, 35]. The reader is referred to a comprehensive literature review [5] and extensive surveys [3, 16, 37] on this topic of deep learning on graphs. Recently, GCNs have been extended to hypergraphs [10, 38]. Below, we give an overview of related research in link prediction on hypergraphs where relationships go beyond pairwise.

Link Prediction on hypergraphs: Machine learning on hypergraphs was introduced in a seminal work [46] that generalised the powerful methodology of spectral clustering to hypergraphs and further inspired algorithms for hypergraph embedding and semi-supervised classification of vertices.

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* [21] and to predict metadata information such as tags, groups, labels, users for entities (images from Flickr) [1]. Techniques for hyperlink prediction on social networks include ranking for link proximity information [21], matrix completion on the (incomplete) incidence matrix of the hypergraph [1, 28], hyperpath-based method [17], and Laplacian tensor methods for context-aware recommendation [41]. These methods are restricted to uniform hypergraphs in which each hyperlink contains the same number of vertices.

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 space [44]. CMM uses expectation maximisation algorithm for optimisation for hyperlink prediction tasks such as predicting missing reactions of organisms' metabolic networks. Recently, n-way interactions have been helpful for link prediction esp. for larger datasets [8]. For a hypergraph, it has been shown that an n-tuple-wise similarity function cannot be a linear function [4, 31, 45]. Recently neural methods for knowledge hypergraph complemetion have achieved impressive results [9]. Knowledge hypergraphs contain hyperlinks in which vertices are ordered. Our paper, on the other hand, is on directed hypergraphs in which hyperlinks are ordered.

Non-Linear Laplacians for Hypergraphs and Digraphs: Recent research has extended the notion of graph Laplacian to directed graphs [39] and hypergraphs through diffusion processes for undirected [7, 27], directed [6, 22, 42], inhomogeneous [23], and submodular [24, 40] hypergraphs. Our proposed method is inspired by these notions of non-linear Laplacians.

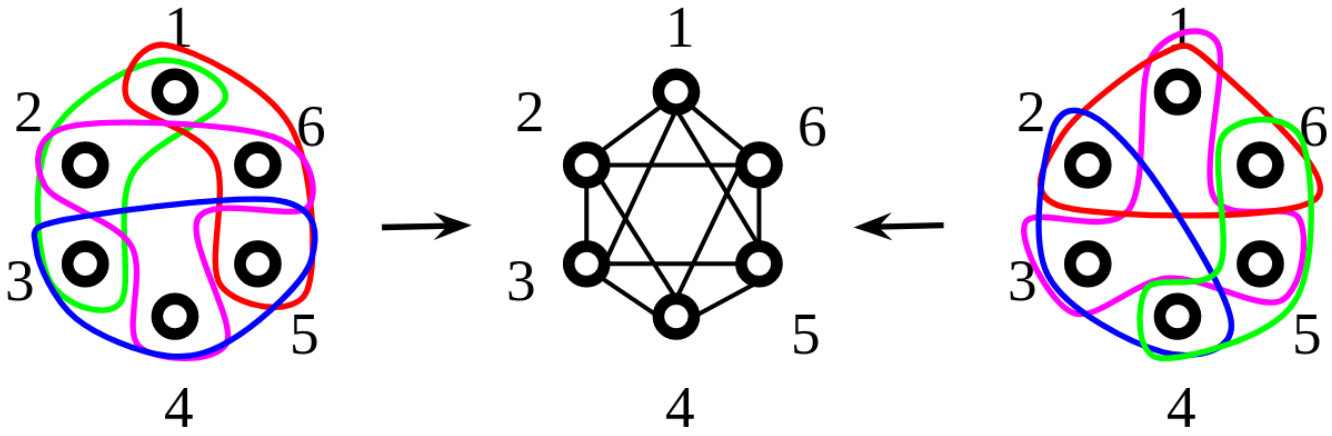

Figure 2: (best seen in colour) An example of two different hypergraphs resulting in the same weighted clique expansion. Both hypergraphs have 6 vertices and 4 hyperedges each. The hypergraph on the left side consists of $\{1,2,3\}, \{1,5,6\}, \{3,4,5\}, \{2,4,6\}$ as its hyperedges. The hypergraph on 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 4 for more details.

3 SETUP

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

3.1 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 vertices potentially forming a link).

In some practical cases such as chemical reaction prediction, there is no need to consider all the hyperlinks in $\bar{E}$ as most of them can be easily filtered out [44]. However in many other cases such as knowledge base canonicalisation, and collaboration networks, there is no notion of "feasible facts" and "feasible collaborations" respectively. The hyperlink prediction problem [44] in such cases is much more challenging. We tackle this challenging problem by proposing a framework that can handle unseen hyperlinks at test time.

Formally, a hyperlink prediction problem [44] 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 co-ordinated matrix minimisation (CMM) algorithm [44] 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.

3.2 Directed hyperlink prediction

A directed hypergraph [12] is an ordered pair $H = (V, E)$ where $V = \{v_1, \cdots, v_n\}$ is a set of n vertices and

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

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.

4 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.

4.1 NHP-U

NHP consists of a trainable hyperlink-aware GCN layer and a hyperlink scoring layer to preserve the higher-order relationships among the vertices in each hyperlink. It is then optimised by a ranking objective in which scores of existing hyperlinks are ranked higher than those of non-existing vertex subsets. A schematic view of NHP-D is shown in Figure 3 and that of NHP-U is shown in 4.

Hyperlink-aware GCN layer: One of the main challenges that arises when working with hypergraphs is the variable cardinality of hyperlinks, i.e., each hyperlink connects arbitrary number of vertices. This challenge is typically handled by approximating the

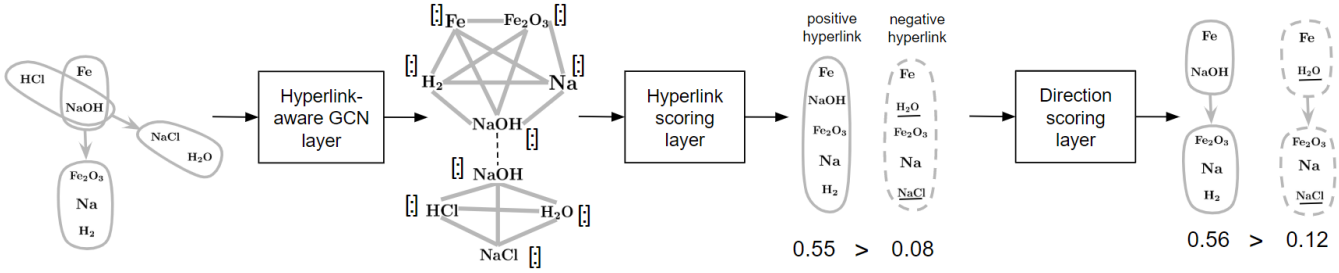

Figure 3: NHP-D for hyperlink prediction. The Hyperlink-aware GCN layer takes the input hypergraph and converts it to its clique expansion (all vertex pairs in each hyperlink are connected by pairwise edges). A GCN is then used to get hyperlink-aware embeddings of vertices (NaOH will have two embeddings). The hyperlink scoring layer assigns a score to each hyperlink. The direction scoring layer then assigns a score to the direction in each link. A ranking mechanism is used in each of the scoring layers in which existing links are ranked higher than non-existing sets of vertices (figure shows an example). Please see the section entitled NHP: Neural Hyperlink Predictor for details.

hypergraph by its clique expansion in which pairwise edges are introduced between all vertex pairs in the hyperlink [4, 10]. Because of the pairwise connections, each hyperlink may not be considered 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 [23].

We propose to address the above issue by refining the embeddings with a GCN layer on the subgraph obtained from the clique expansion of each hyperlink and then passing the embeddings to a trainable hyperlink scoring layer. 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 [13]:

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

where x_v are initial features for GCN. The initial features could be any available additional features (e.g. bag-of-words for documents). In our work if the initial features are not available, we use the unsupervised node2vec [14] algorithm for initialisation (on the clique expansion of the hypergraph). We then pass the set of refined embeddings $\{h_v^{(e)} : v \in e\}$ for each hyperlink $e \in E$ to a hyperlink scoring function described below.

Hyperlink scoring 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 hyperlink scoring function, I_e, to preserve the higher-order relationships in the input hypergraph. The function takes the form

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

where W is a parameter of dimension $1 \times d$ and σ is the sigmoid function. Intuitively, the 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 g. Thus, a natural choice of g is to use the mean of the embeddings of a hyperlink e 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 *explicitly* guarantee any priors on the vertex embeddings of a hyperlink. In hypergraphs such as chemical reaction networks, it is often the case that vertices (chemical substances) with very different/dissimilar and complementary properties react and form a hyperlink. For example, acids react with bases and form a hyperlink (to give salt and water). On the other hand, in some other hypergraphs such as co-authorship networks, it is often the case that vertices (authors of a paper) with similar properties (such as research interests) collaborate to form a hyperlink. Ideally, we would like g and the resulting optimisation to encode these priors in a task-specific fashion. Based on the above observations, we propose the following:

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

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 hyperlink scoring function is precisely what makes NHP handle unseen links at test time.

Optimisation: Hyperlinks in the input hypergraph represent known relationships among the vertices of the hyperlink. The set of unknown relationships i.e., $2^V - E$ may, in fact, contain undiscovered hyperlinks and belong to the existing ones. Following prior work [26], 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

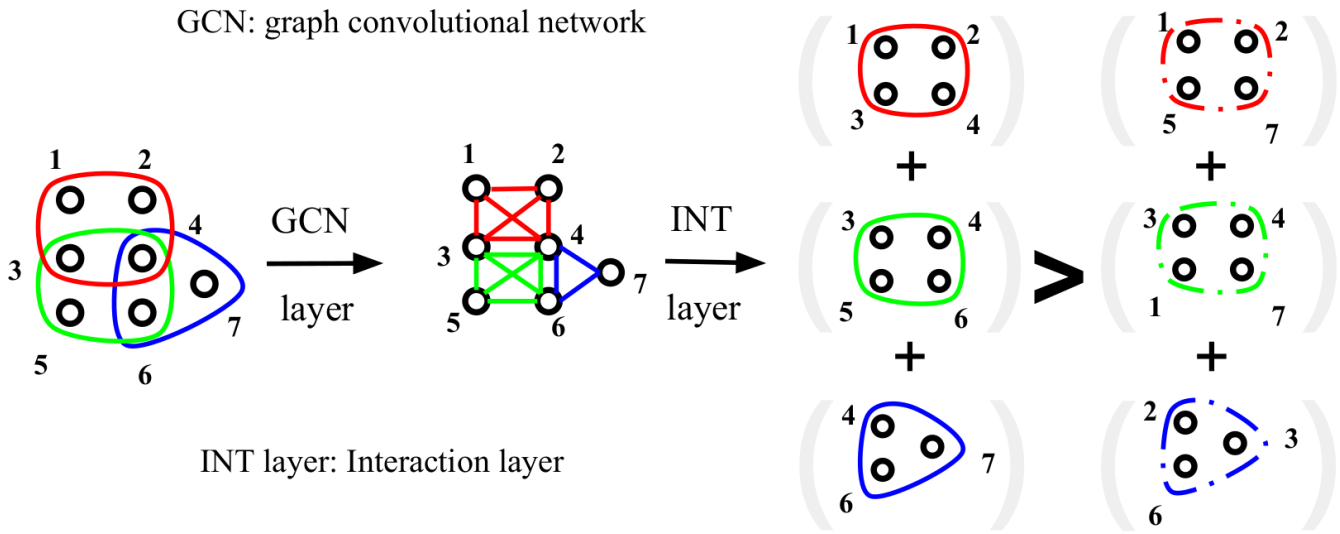

Figure 4: (best seen in colour) NHP for undirected hyperlink prediction. The input hypergraph is converted to its clique expansion to get the initial embeddings on the vertices of the hypergraph. The initialisations are then fed to a GCN layer that refines the embeddings on the subgraph obtained from 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.

logistic function $\Lambda(x) = \log(1 + e^x)$. The loss $\mathcal{L}$ above tries to maximise the number of hyperlink scores (in E) that are higher than the average score of the unknown vertex sets in F. It ranks the observed hyperlinks above the unobserved ones. All weights of NHP-U i.e W_{GCN}, W are learned end-to-end using stochastic gradient descent. We motivate our choice of I_e and I_f (defined in Equation 4) in Equation 6 through the following proposition.

Proposition 1. *Let $G = (V, E)$ be a hypergraph with a sampled negative hyperlink set F. For an arbitrary set of vertices $S \in V$, define I_S as in Equation 4, and let $h_v^{(S)}$, $v \in V$ represent the GCN embeddings as obtained from Equation 1. For simplicity, assume $d = 1$ and $b = 0$. Define $M_S := \text{maxmin}\{h_v^{(S)}\}_{v \in S}$ where* maxmin *is as defined in Equation 5. Then, for a hyperlink $e \in E$,*

$$\left(\frac{1}{|F|}\sum_{f \in F} I_f\right) - I_e \leq 0$$

when any one of the following sufficient conditions is satisfied:

(1) $I_e = \sigma\left(W \cdot \max_{S \in 2^V} M_S\right)$ *and* $W \geq 0$
(2) $I_e = \sigma\left(W \cdot \min_{S \in 2^V} M_S\right)$ *and* $W < 0$

Proof. The proof follows directly from the definitions of max and min and the monotonicity of the sigmoid function σ. □

The proposition can be easily extended to the cases when $d \geq 1$ and/or $b \neq 0$. The proposition essentially implies that in extreme cases there exists:

(1) A non-negative weight that minimises the loss for hyperlinks in which very dissimilar vertices interact (e.g. acid + base)
(2) A negative weight that minimises the loss for hyperlinks in which similar vertices interact (e.g. author collaboration)

Hence, our proposed scoring function adaptively learns weights in a task-specific manner and includes more prior knowledge than the simple mean of the embeddings. It is worth pointing out that the maxmin function has, in spirit, been used to define non-linear Laplacians for hypergraphs [7, 27]. The essence of Laplacian operators is smoothing based on certain topology (similarity of vertices). However, in our work we show that coupled with a ranking objective, the maxmin function can flexibly be used when two dissimilar vertices interact to form a hyperlink (e.g. acid + base). We now give our method to sample the negative set F.

Sampling method: 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.

4.2 NHP-D

In this section, we discuss how the proposed NHP can be extended to link prediction in directed hypergraphs. NHP-D consists of three components two of which are hyperlink-aware embedding, and a

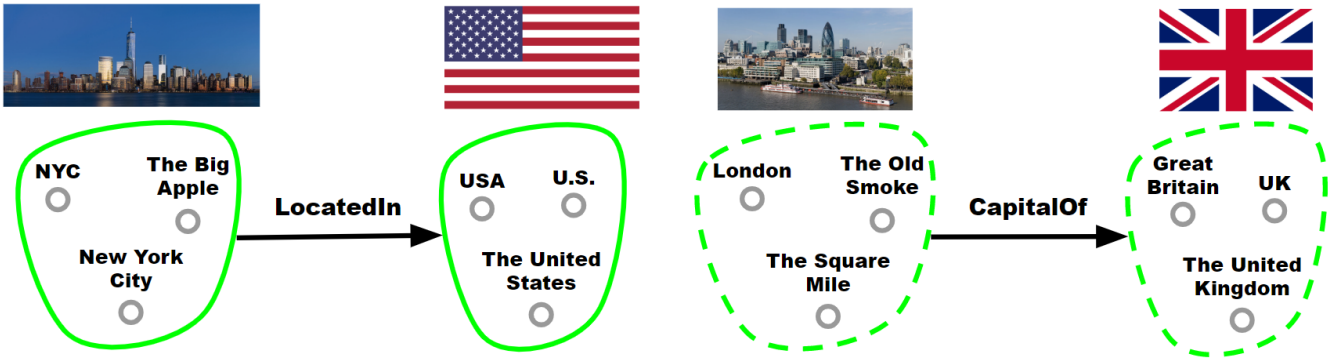

Figure 5: (best seen in colour) Modelling a canonicalised knowledge graph as a multi-relational directed hypergraph. The vertices of the hypergraph are noun phrases (surface forms). Each labelled directed hyperedge consists of a tail hyperlink representing the canonicalised subject entity, a head hyperlink representing the canonicalised object entity and the label (e.g. LocatedIn) indicating the relationship between the subject and the object. The link prediction problem is essentially a canonicalised knowledge graph completion problem. Please see Section 5 for more details

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	30956	11433	2253	2084
number of vertex features	382	3763	298	26	26

hyperlink scoring function. The third 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 hyperlink-aware embedding as described for NHP-U. For the hyperlink scoring function, we use the difference between the element-wise maximum in tail and minimum in head hyperlinks. 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 $m \leq 0$. The maxmin_+ function is, in spirit, used to define Laplacians for directed hypergraphs [6, 42]. Our proposition 1 can be easily extended to the directed case and hence is useful even when dissimlar vertices interact to form a hyperlink.

4.3 Direction scoring layer:

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.

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

We use a joint optimisation strategy to get the hyperlink 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 i.e. W_{GCN}, W, W_{BL} are all updated end-to-end using backpropagation.

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.

4.4 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 hyperlink scoring 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 $O(N_1 + N_2 + |E|)$.

dataset →	iAF1260b		iJO1366		USPTO	
model ↓	AUC	Recall@k	AUC	Recall@k	AUC	Recall@k
HGNN [10]	0.55 ± 0.02	0.23 ± 0.05	0.53 ± 0.02	0.25 ± 0.01	0.59 ± 0.01	0.22 ± 0.04
HyperGCN [38]	0.55 ± 0.03	0.24 ± 0.09	0.52 ± 0.04	0.26 ± 0.03	0.59 ± 0.01	0.23 ± 0.03
Hyper-SAGNN [45]	0.53 ± 0.02	0.20 ± 0.03	0.51 ± 0.02	0.23 ± 0.02	0.56 ± 0.02	0.21 ± 0.03
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$_+$	**0.58 ± 0.02**	**0.26 ± 0.04**	**0.56 ± 0.01**	**0.28 ± 0.03**	**0.63 ± 0.02**	**0.25 ± 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
HGNN	0.76 ± 0.03	0.40 ± 0.05
HyperGCN	0.77 ± 0.02	0.39 ± 0.06
Hyper-SAGNN	0.74 ± 0.03	0.39 ± 0.03
node2vec	0.57 ± 0.01	0.38 ± 0.04
node2vec-GCN	0.62 ± 0.02	**0.41 ± 0.03**
node2vec-mean	0.65 ± 0.02	0.41 ± 0.05
node2vec-maxmin$_+$	0.75 ± 0.01	0.40 ± 0.04
NHP-D-mean	0.83 ± 0.03	0.40 ± 0.03
NHP-D-maxmin$_+$	**0.85 ± 0.04**	**0.43 ± 0.03**

Table 3: Mean AUC and Recall@k values on Reverb45k.

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.

5 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. We discuss the motivation for them.

5.1 Canonicalising an open knowledge graph

Open information extraction methods have helped in construction of large open knowledge graphs (KGs) from text [11, 32]. The extracted noun phrases are not canonicalised, leading to redundant and ambiguous facts. For example, <New York City, CityIn, U.S.> and <NYC, CityIn, U.S.> are two triples representing the same fact. When we query an open KG for facts about an entity by one name, there is no guarantee that we get all facts about the entity. Hence, there is a need to canonicalise noun phrases i.e. identify clusters of noun phrases representing the same entity [11, 32].

Figure 5 shows how an entity canonicalised knowledge graph can be modelled by a multi-relational directed hypergraph. A multi-relational directed hypergraph is a labelled directed hypergraph (V, E, L) in which V is the set of vertices and

$$E = \{(t_1, h_1, l_1), \cdots, (t_m, h_m, l_m)\} \subseteq 2^V \times 2^V \times L$$

is the set of labelled directed hyperlinks with labels $l_1, \cdots, l_m \in L$ in the label set.

In this work, we approximate an entity-canonicalised knowledge graph by an unlabelled directed hypergraph in which, for each (subject, relation, object) triple, the canonicalised subject entity represents a tail hyperlink, the canonicalised object entity represents the corresponding head hyperlink, and the existence of a relation is captured by a direction from tail to head. We treat all symmetric relations in such a knowledge graph by having directions on both sides.

The link prediction task would be to complete a canonicalised knowledge graph, i.e., given a bunch of canonicalised entities and directions (relationships) between canonicalised pairs, we aim to predict undiscovered canonicalised entities and relationships between pairs. We used Reverb45k used in previous works [11, 32].

5.2 Reactions of chemical reaction networks

Prediction of reaction outcomes is a fundamental problem in computational chemistry [18]. While products can be determined for simple reactions, it is a major challenge for many complex organic reactions. Chemical reaction experimentation is time consuming, expensive, and requires a domain expert. This motivates the chemical reaction prediction problem in which given a bunch of known reactions, we predict (potentially complex) unknown reactions.

We used the USPTO dataset [18] as an organic reaction dataset in our experiments. We used a subset of chemical substances containing only CHNOPS (carbon, hydrogen, nitrogen, oxygen, phosporous, sulphur) to get a reasonably dense hypergraph. Prior work [18] provides product templates for each reactant. We use these templates to get candidate reactions.

Reconstructed metabolic networks are important tools for understanding the metabolic basis of human diseases, increasing the yield of biologically engineered systems, and discovering novel drug targets. Some vital reactions can be missing from a metabolic network, which can severely impair their utility [30]. Thus, it is critical to develop computational methods for completing metabolic networks. We used the two largest datasets used in a prior work [44] and their candidate reactions.

5.3 Collaborations in a co-authorship network

Research collaborations in scientific community have been extensively studied to understand team dynamics in social networks [2]. Co-authorship data provide a means to analyse research collaborations. We used DBLP[2] for co-authorship data. We used a

[2]https://aminer.org/citation

dataset →	iAF1260b		iJO1366		USPTO	
model ↓	AUC	Recall@k	AUC	Recall@k	AUC	Recall@k
HGNN [10]	0.62 ± 0.02	0.27 ± 0.04	0.61 ± 0.03	**0.32 ± 0.02**	0.69 ± 0.02	0.26 ± 0.01
HyperGCN [38]	0.62 ± 0.02	0.28 ± 0.02	0.60 ± 0.02	**0.32 ± 0.02**	0.70 ± 0.03	0.29 ± 0.03
Hyper-SAGNN [45]	0.60 ± 0.01	0.26 ± 0.01	0.56 ± 0.03	0.30 ± 0.04	0.67 ± 0.01	0.27 ± 0.01
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
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
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
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
NHP-U-mean	0.60 ± 0.04	0.28 ± 0.06	0.61 ± 0.02	0.29 ± 0.02	0.65 ± 0.02	0.33 ± 0.05
NHP-U-maxmin	**0.64 ± 0.03**	**0.31 ± 0.03**	**0.63 ± 0.02**	**0.32 ± 0.02**	**0.74 ± 0.02**	0.37 ± 0.02
SHC	0.65 ± 0.01	0.31 ± 0.02	0.64 ± 0.01	0.33 ± 0.02	0.56 ± 0.01	0.22 ± 0.01
CMM	0.64 ± 0.04	0.30 ± 0.14	0.64 ± 0.03	0.35 ± 0.10	0.68 ± 0.01	**0.37 ± 0.01**

Table 4: Mean AUC and Recall@k values (higher is better) for link prediction in undirected hypergraphs on the three chemical reaction datasets. Our method is competitive with SHC and CMM even though our method does not use candidate hyperlinks during training.

model	AUC	Recall@k
HGNN	0.65 ± 0.04	0.33 ± 0.03
HyperGCN	0.66 ± 0.01	0.32 ± 0.02
Hyper-SAGNN	0.65 ± 0.01	0.34 ± 0.04
node2vec	0.56 ± 0.03	0.30 ± 0.03
node2vec-GCN	0.59 ± 0.03	0.32 ± 0.04
node2vec-mean	0.58 ± 0.01	0.31 ± 0.03
node2vec-maxmin	0.63 ± 0.04	0.35 ± 0.06
SHC	**NA**	**NA**
CMM	**NA**	**NA**
NHP-U-mean	0.56 ± 0.02	0.28 ± 0.04
NHP-U-maxmin	**0.69 ± 0.02**	**0.38 ± 0.05**

Table 5: Mean (± std) Recall@k values on the DBLP dataset. NA: not applicable. SHC and CMM cannot handle unseen hyperlinks at test time.

subset of papers published in only artificial intelligence conferences (AAAI, IJCAI, NeurIPS, ICML, CVPR, ICCV, ACL, NAACL, etc.). A co-authorship hypergraph contains each author as a vertex and each paper represents a hyperlink connecting all the authors of the paper. The hyperlink prediction problem is given a set of collaborations, to essentially predict other potential collaborations.

We now discuss the experiments of our proposed NHP.

6 EXPERIMENTS

We used a knowledge base (KB), 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 KB as a directed hypergraph and co-authorship data as an undirected hypergraph for a total of 4 undirected and 4 directed hypergraph data.

On all the real-world datasets, we report mean AUC and Recall@k (k is half the number of missing links) numbers on the held-out test split averaged over 10 random splits of train, validation, and test set of hyperlinks. 20% hyperlinks in each dataset was used for training (for 50 epochs), 10% for validation, and the remaining 70% held-out for testing the proposed models and all the baselines. Hyper-parameters were optimised using grid-search.

6.1 Experiments on directed hypergraphs

Since there are no proposed approaches for hyperlink prediction in directed hypergraphs, we modify existing embedding methods as baselines for the problem:

- **HGNN [10]:** This is a clique expansion-based approach for hypergraphs. We used their formulation instead of our GCN layer of Equation 1.
- **HyperGCN [38]:** This method uses non-linear Laplacian to define GCNs on hypergraphs.
  Since the above two methods are not proposed for link prediction, we used our proposed maxmin$_+$ hyperlink scoring layer for hyperlink prediction with these methods.
- **Hyper-SAGNN [45]**: This is a recently proposed self-attention-based approach for the problem. Since it is not proposed for directed hypergraphs, we used our proposed Equations 7 and 8 to optimise it.
- **node2vec:** we used node2vec embeddings with the following hyperlink scoring 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 hyperlink.
- **node2vec-GCN:** we used node2vec embeddings as initialisation to a GCN and then train the GCN's parameters with the previously mentioned scoring function. Note that this baseline does not use the proposed hyperlink scoring functions viz., mean and maxmin$_+$.
- **node2vec-mean:** we used node2vec embeddings as initialisation to a trainable scoring function which takes the mean of the vertex embeddings to learn the weights

- **node2vec-maxmin$_+$:** we used node2vec embeddings as initialisation to a scoring 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 scoring layer with all the aforementioned baselines and learn the parameters W_{BL}. The results are shown in Table 2. 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 relationships 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 {New York, London}) and head containing {UK, Brazil}). The set of 10% missing hyperlinks union these set of negative hyperlinks is test set.

Statistical test. We performed a Welch t-test [36] 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.

6.2 Experiments on undirected hypergraphs

The state-of-the-art methods for link prediction in undirected hypergraphs are Co-ordinated matrix minimisation (CMM) [44], and Spectral Hypergraph Clustering (SHC) [46]. We compared NHP-U against the following baselines:

- **CMM [44]:** This baseline uses the expectation-maximisation algorithm in the adjacency space to predict hyperlinks.
- **SHC [46]:** 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.
  We note that these two methods (CMM, SHC) are not embedding-based methods and hence cannot be used to predict hyperlinks in *directed hypergraph experiments*.

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) [44], and Spectral Hypergraph Clustering (SHC) [46], require all candidate hyperlinks (for reaction datasets) to be present during training. 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 hyperlink scoring function is trainable from the input hypergraph.

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 is not used).

Hyperparameter	Values
node2vec embedding size	128
GCN hidden size	{4, 8, 16, 32, 64, 128, 256, 512}
λ	{0.001, 0.01, 0.1, 1, 10, 100, 1000}
learning rate	0.001

Table 6: : Hyperparameters used in the experiments

Value	DBLP	USPTO	Reverb45k
4	0.60 ± 0.03	0.54 ± 0.02	0.73 ± 0.05
16	0.63 ± 0.02	0.59 ± 0.03	0.79 ± 0.05
64	0.66 ± 0.01	0.60 ± 0.05	0.80 ± 0.04
256	0.68 ± 0.05	0.61 ± 0.03	0.83 ± 0.02
512	**0.69 ± 0.02**	**0.63 ± 0.02**	**0.85 ± 0.04**

Table 7: : Parameter sensitivity of hidden size on DBLP, USPTO, and Reverb45k. We report AUC in the table.

NHP-U, however, uses a ranking objective and the hyperlink scoring function 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.

6.3 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, CMM are not). All were run on a GeForce GTX 1080 Ti GPU machine.

6.4 Hyperparameters

Hyper-parameters such as GCN hidden size and λ were optimised using grid-search. The set of values and the optimal values for these hyperparameters are also shown in Table 6. Dropout, learning rate, weight decay were fixed to the values of a prior work [19].

We also show the sensitivity of GCN hidden size on the USPTO dataset in Table 6. We can conclude that the performance of NHP is relatively stable within a range of embedding dimensions, and the performance drops when embedding dimension is too small.

6.5 Ablation Study

Tables 2, 3, 4, and 5 showed that all the components of our model are required to achieve effective results. We also compared randomly initialised embeddings and node2vec embeddings on USPTO undirected hypergraph. random-maxmin achieved an AUC of 0.62 ± 0.04 with a Recall of 0.23 ± 0.02 while node2vec-maxmin achieved an AUC of 0.68 ± 0.01 and a Recall of 0.26 ± 0.01. This also shows that the initial features are important for effective performance.

7 CONCLUSION AND FUTURE WORK

We have introduced NHP, a novel approach for hyperlink prediction for both undirected and the first method on directed hypergraphs. The novel scoring functions make no assumptions on the type of interactions within

hyperlinks. NHP can effectively handle unseen hyperlinks at test time. In future, we explore multi-relational directed hypergraphs in which hyperedges have labels and separately multi-modal datasets (both shown in Figure 5).

ACKNOWLEDGEMENT

This work is supported in part by the Ministry of Human Resource Development (Government of India). Anand Louis was supported in part by SERB Award ECR/2017/003296 and a Pratiksha Trust Young Investigator Award. We give credits to Google Images and Wikipedia for the pictures used in the paper.

REFERENCES

[1] Devanshu Arya and Marcel Worring. 2018. Exploiting Relational Information in Social Networks Using Geometric Deep Learning on Hypergraphs. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR)*. 117–125.

[2] Haoli Bai, Zhuangbin Chen, Michael R. Lyu, Irwin King, and Zenglin Xu. 2018. Neural Relational Topic Models for Scientific Article Analysis. In *CIKM*.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR, arXiv:1806.01261* (2018).

[4] Inci M Baytas, Cao Xiao, Fei Wang, Anil K. Jain, and Jiayu Zhou. 2018. HHNE: Heterogeneous Hyper-Network Embedding. In *IEEE International Conference on Data Mining (ICDM)*. 875–880.

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* (2017).

[6] T.-H. Hubert Chan, Zhihao Gavin Tang, Xiaowei Wu, and Chenzi Zhang. 2019. Diffusion operator and spectral analysis for directed hypergraph Laplacian. *Theor. Comput. Sci. (TCS)* 784 (2019), 46–64.

[7] T.-H. Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. 2018. Spectral Properties of Hypergraph Laplacian and Approximation Algorithms. *J. ACM* (2018).

[8] Se eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. 2020. How Much and When Do We Need Higher-order Information in Hypergraphs? A Case Study on Hyperedge Prediction. In *Proceedings of the 29th International Conference on World Wide Web (WWW)*.

[9] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2020. Knowledge Hypergraphs: Prediction Beyond Binary Relations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*.

[10] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph Neural Networks. In *Proceedings of the Thirty-Third Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 3558–3565.

[11] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M. Suchanek. 2014. Canonicalizing Open Knowledge Bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. 1679––1688.

[12] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed Hypergraphs and Applications. *Discrete Appl. Math.* (1993).

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

[14] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 855––864.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* (2017).

[17] Jie Huang, Xin Liu, and Yangqiu Song. 2019. Hyper-Path-Based Representation Learning for Hyper-Networks. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*.

[18] Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. 2017. Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network. In *NIPS*.

[19] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[20] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* (2011).

[21] Dong Li, Zhiming Xu, Sheng Li, and Xin Sun. 2013. Link Prediction in Social Networks Based on Hypergraph. In *WWW*.

[22] Pan Li, Niao He, and Olgica Milenkovic. 2018. Quadratic Decomposable Submodular Function Minimization. In *Advances in Neural Information Processing Systems (NeurIPS) 31*. Curran Associates, Inc., 1054–1064.

[23] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous Hypergraph Clustering with Applications. In *Advances in Neural Information Processing Systems (NeurIPS) 30*. Curran Associates, Inc., 2308–2318.

[24] Pan Li and Olgica Milenkovic. 2018. Submodular Hypergraphs: p-Laplacians, Cheeger Inequalities and Spectral Clustering. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 3014–3023.

[25] David Liben-Nowell and Jon Kleinberg. 2003. The Link Prediction Problem for Social Networks. In *CIKM*.

[26] Yashu Liu, Shuang Qiu, Ping Zhang, Pinghua Gong, Fei Wang, Guoliang Xue, and Jieping Ye. 2017. Computational Drug Discovery with Dyadic Positive-Unlabeled Learning. In *Proceedings of the 2017 SIAM International Conference on Data Mining*.

[27] Anand Louis. 2015. Hypergraph Markov Operators, Eigenvalues, Approximation Algorithms. In *STOC*.

[28] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. In *NIPS*.

[29] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* (2016).

[30] Vinay Satish Kumar, Madhukar S. Dasika, and Costas D. Maranas. 2007. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics* (2007).

[31] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2018. Structural Deep Embedding for Hyper-Networks. In *Proceedings of the Thirty-Third Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 426–433.

[32] Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. CESI: Canonicalizing Open Knowledge Bases Using Embeddings and Side Information. In *Proceedings of the 2018 World Wide Web Conference (WWW)*. 1317—-1327.

[33] Shikhar Vashishth, Prateek Yadav, Manik Bhandari, and Partha Talukdar. 2019. Confidence-based Graph Convolutional Networks for Semi-Supervised Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[35] Petar Veličković, William Fedus, William L. Hamilton, Pietro Lió, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. In *International Confernece on Learning Representations (ICLR)*.

[36] B. L. Welch. 1947. The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. *Biometrika* 34, 1/2 (1947), 28–35. http://www.jstor.org/stable/2332510

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR, arXiv:1901.00596* (2019).

[38] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS) 32*. Curran Associates, Inc., 1509–1520.

[39] Yuichi Yoshida. 2016. Nonlinear Laplacian for Digraphs and Its Applications to Network Analysis. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*. 483–492.

[40] Yuichi Yoshida. 2019. Cheeger Inequalities for Submodular Transformations. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2582–2601.

[41] Chia-An Yu, Ching-Lun Tai, Tak-Shing Chan, and Yi-Hsuan Yang. 2018. Modeling Multi-Way Relations with Hypergraph Embedding. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. 1707—-1710.

[42] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and T-H. Hubert Chan. 2017. Re-revisiting Learning on Hypergraphs: Confidence Interval and Subgradient Method. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 4026–4034.

[43] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS) 31*. Curran Associates, Inc., 5171–5181.

[44] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. 2018. Beyond Link Prediction: Predicting Hyperlinks in Adjacency Space. In *Proceedings of the Thirty-Second Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 4430–4437.

[45] Ruochi Zhang, Yuesong Zou, and Jian Ma. 2020. Hyper-{SAGNN}: a self-attention based graph neural network for hypergraphs. In *International Confernece on Learning Representations (ICLR)*.

[46] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NeurIPS)*.