Neural Message Passing for Multi-Relational Ordered and Recursive Hypergraphs

Naganand Yadati
naganand@iisc.ac.in
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, Karnataka, 560012

Abstract

Message passing neural network (MPNN) has recently emerged as a successful framework by achieving state-of-the-art performances on many graph-based learning tasks. MPNN has also recently been extended to multi-relational graphs (each edge is labelled), and hypergraphs (each edge can connect any number of vertices). However, in real-world datasets involving text and knowledge, relationships are much more complex in which hyperedges can be multi-relational, recursive, and ordered. Such structures present several unique challenges because it is not clear how to adapt MPNN to variable-sized hyperedges in them. In this work, we first unify existing MPNNs on different structures into G-MPNN (Generalised-MPNN) framework. Motivated by real-world datasets, we then propose a novel extension of the framework, MPNN-R (MPNN-Recursive) to handle recursively-structured data. Experimental results demonstrate the effectiveness of proposed instances of G-MPNN and MPNN-R.

1 Introduction

Message passing neural network (MPNN) has recently emerged as a successful framework by achieving state-of-the-art performances on many graph-based learning tasks [21]. The message-passing operation in the MPNN framework can be viewed as recursive neighbourhood aggregation, where local neighbourhood messages are aggregated and passed on to neighbouring vertices. MPNN has also recently been extended to multi-relational graphs in which each edge is labelled (and possibly directed), and separately to hypergraphs in which each edge can connect any number of vertices. However, in real-world datasets involving text and knowledge, relationships are much more complex in which hyperedges can act as vertices recursively in other hyperedges. Hyperedges can also be multi-relational with vertices appearing in a fixed order. We illustrate such structures with a toy example in Figure[1]. Multi-relational ordered hypergraphs have been shown to provide more flexible organisation of multi-ary relational facts than multi-relational directed edges and have been a recent research topic of interest [66,16]. Recursive hypergraphs [47] have been shown to flexibly represent a few sentence types such as claims about claims in natural language (e.g. A claimed that B claimed C). Recursively structured hypergraphs are also seen in academic network datasets. Such structures present several unique challenges because it is not clear how to adapt MPNN to variable-sized hyperedges in them. Our contributions can be summarised as follows:

- We provide a unified MPNN-style framework, which we call G-MPNN (Generalised-MPNN), for multi-relational ordered hypergraphs. Several notable examples of models can be described using the unified framework
- Motivated by real-world datasets, we explore the unexplored problem of inductive vertex embedding (embedding unseen vertices at test time) in multi-relational ordered hypergraphs.
We demonstrate the strong inductive capability of G-MPNN on real-world multi-relational ordered hypergraph datasets.

- Motivated by recursively-structured datasets, we propose a novel extension of MPNN, termed MPNN-R (MPNN-Recursive) and show its effectiveness on real-world datasets.

## 2 Related Work

In this section, we discuss related work. In particular, we discuss relevant work on MPNNs, and their explorations on multi-relational graphs (including on the closely related multiplex networks, and heterogeneous graphs), and hypergraphs.

### 2.1 Message-Passing Neural Networks (MPNNs)

MPNNs were originally proposed as a framework for deep learning on graphs [21]. MPNN has inspired the current state-of-the-art techniques for graph representation learning (GRL). The reader is referred to comprehensive reviews [7] and extensive surveys [27] on this topic of GRL. The message-passing operation in the MPNN framework can be viewed as recursive neighbourhood aggregation, where local neighbourhood messages are aggregated and passed on to neighbouring vertices [18]. Notable instances of the MPNN framework include popular graph neural networks such as Graph Convolutional Networks (GCNs) [36], ChebNet [14], GraphSAGE [26], Graph Attention Networks [60], Neural Fingerprints [15], Gated Graph Sequence Neural Networks (GGNN) [42], Graph Isomorphism Networks [68], etc. GNNs (and MPNNs) came into existence thanks to two seminal publications on convolutional [8] and recurrent [52] neural networks on graphs. The MPNN framework has been extended to multi-relational graphs in several ways which we discuss next.

### 2.2 MPNNs on Multi-Relational Graphs

The earliest attempts at extending GNNs to multi-relational graphs propose relation-specific parameters and include relational GGNN [25], and relational GCNs [53] [56]. Recent attempts include learning relation-specific weights [54], neighbour and logic-based attention [63], triple-based attention [48], hierarchical attention [29], and relation and neighbour-based attention [45]. The state-of-the-art MPNN models include aggregating neighbour embeddings through relation embedding composition [59], and dynamically pruned MPNN on input-dependent subgraphs [69]. GCNs with direction-specific parameters have become quite popular approximations in natural language processing on syntactic multi-relational graphs [45] [55] [58].
Multiplex Networks: Another closely related line of research extends MPNN to multiplex networks or multi-view networks. Attempts along this line include effectively combining the Laplacians of the multiple views [44][41], a self-attention-based approach [9], and an unsupervised embedding method [49] that extends Deep Graph Infomax [61].

Heterogeneous Graphs: A related line of research is MPNN extension to heterogeneous graphs in which vertices (and edges) are typed. Ideas successfully used along this line include typed attention [43], neighbour attention [73], vertex-level and semantic-level attention [65], metapath-based aggregation [19], and vertex, edge-type dependent parameterisation [29]. On a related note, MPNN originally incorporated edge features as also done by three recent works [22][12][64].

2.3 MPNNs on Hypergraphs

Hypergraphs are challenging data structures as they encode relationships going beyond pairwise associations [81]. Hypergraph Neural Network [17] approximates the hypergraph by its clique expansion [1] and applies traditional graph-based deep approaches such as GCN [11] on it. The clique expansion has been used subsequently in label propagation network [77], hyperedge prediction [5], and at least a couple of neural hypergraph construction methods [30][31].

Hypergraph Convolutional Network [70] uses the mediator-based hypergraph Laplacian [10] which is one of many non-linear Laplacians [28][72][40][37][39] and shows improvements on hypergraphs with noisy hyperedges. Powerset Convolutional Networks [67] uses tools from discrete signal processing to principally define convolution on set functions. Hyper-SAGNN [75] proposes a self-attention-based approach for hyperedge prediction.

Our contributions: In all the publications that we have seen above, (hyper)edges do not participate as vertices in other (hyper)edges recursively and also are not multi-relational ordered. Our contributions are precisely to address these limitations of existing MPNN-based approaches.

3 Generalising MPNN to Multi-Relational Ordered hypergraphs

In this section, we will first see all the notations used. We will then briefly see MPNN before our proposed unified MPNN for multi-relational ordered hypergraphs.

3.1 Notations

We represent a multi-relational ordered hypergraph as a quadruple, $\mathcal{H} = (V, E, P, R)$. Here, $V$ is a set of $n$ vertices. $E = \{e_1, \cdots, e_m\}$ is a multiset of hyperedges with each hyperedge $e \in E$ satisfying $e \subseteq V$. The notation $P$ is used to denote $P := \{P_e : e \in E\}$ where each $P_e$ is a positional mapping $P_e : e \rightarrow \{1, \cdots, p\}$, with $p$ an integer and $p \leq n$. We note that there can exist integers $i, j$ with $1 \leq i \leq m, 1 \leq j \leq m,$ and $i \neq j$ such that $e_i = e_j$ but $P_{e_i} \neq P_{e_j}$ (e.g. directed graph containing a directed edge in both forward and backward directions).

$R : (E \times P) \rightarrow \{1, \cdots, r\}$ is a relation mapping that maps each (hyperedge, position) pair to one of $r$ pre-defined relations. We see that there can exist $(e_i, P_{e_i})$ and $(e_j, P_{e_j})$ such that $e_i = e_j$ but $R(e_i, P_{e_i}) \neq R(e_j, P_{e_j})$ (e.g. SonOf and FatherOf relationships). For a vertex $v \in V$, let us define $I_v := \{e \in E : v \in e\}$ i.e. $I_v$ is the set of hyperedges incident on $v$.

We now briefly describe MPNN [21] on an undirected graph $G = (V, E)$ with $N_v$ denoting the neighbourhood of vertex $v \in V$, and $e_{vw}$ denoting the edge features of an edge $\{v, w\} \in V$.

3.2 Message Passing Neural Network (MPNN)

The forward propagation of an MPNN has two phases viz., 1) a message passing phase (which runs for $T$ steps), and 2) a readout phase. The message passing phase is defined in terms of the message function $M_t$ and the vertex update function $U_t$ where $t$ is the time step with $t = 1, \cdots, T$. The message and the readout phases are respectively of the form
Table 1: Different existing instantiations of G-MPNN on different structures.

| Structure Type                  | Max. edge size, $\max_{e \in E} |e|$ | # relations, $r$ |
|---------------------------------|----------------------------------|-----------------|
| Graphs [21,36,26,15,42,68]     | 2                               | 1               |
| Multi-relational graphs [25,53] | 2                               | $\geq 1$        |
| Multiplex networks [34,41,9,49] | 2                               | $\geq 1$        |
| Heterogeneous graphs [43,73]    | 2                               | $\geq 1$        |
| Hypergraphs [17,70,30,67]       | $\geq 2$                        | 1               |

**Message:**

$$m_{v}^{t+1} = \sum_{u \in \mathcal{N}_v} M_t(h_u^t, h_v^t, e_{uv}), \quad h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \quad (1)$$

**Readout:**

$$\hat{y} = R(\{ h_v^T : v \in V \})$$

where $M_t, U_t,$ and $R$ are all differentiable functions. We will now see our generalised MPNN framework for multi-relational ordered hypergraphs.

### 3.3 G-MPNN (Generalised-MPNN)

Motivated by rich structures in real-world data (please see related work), we propose a generalised message function for a multi-relational ordered hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{P})$. The key modification is in the message function and is as follows.

$$m_{v}^{t+1} = g\left( \left\{ M_t(h_u^t, \{ (w, h_w^t) \}_{w \in \mathcal{E} - v}, \mathcal{R}(e, P_e), P_e \right\}_{e \in \mathcal{I}_v} \right) \quad (2)$$

where $g$ is any parameterised differentiable function (e.g. element-wise mean, max, sum), $h_v^t$ is the hidden representation of a vertex $v \in \mathcal{V}$ at time step $t$. The vertex update function, and the readout phase remain the same as in Equation (1).

Several notable models in the literature that can be described using the generalised MPNNs of Equation (2) Before describing the specific models, we will first see the following proposition that makes MPNN on a heterogeneous graph a special case of our generalised framework.

**Proposition 1.** Let $G = (\mathcal{V}, \mathcal{E}, S)$ be a heterogeneous graph with $\mathcal{V}$ as a set of vertices, $\mathcal{E}$ as a set of directed edges, and a function $S : \mathcal{V} \rightarrow \{1, \ldots, s\}$ that maps each $v \in \mathcal{V}$ to a type $S_v$ to one of $s$ pre-defined types. Any heterogeneous graph $G = (\mathcal{V}, \mathcal{E}, S)$ is a special $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{P})$ with

- $\mathcal{V} = \mathcal{V}$, and $\mathcal{E} = \{(u, v) : (u, v) \in \mathcal{E}\}$
- $P_e(u) = 1, and P_e(v) = 2$ for each $(u, v) \in \mathcal{E}$ (and $e \in \mathcal{E}$).
- $\mathcal{R}(e, P_e) = (s - 1) * S_u + S_v$ for each $e \in \mathcal{E}$

**Proof.** The key step in the proof is item 3 above where we map the relation associated with each (edge, direction) pair in terms of the vertex types. We note that if $G$ has $s$ vertex types then $\mathcal{H}$ would have $s * (s - 1)$ relations. Each directed edge $(u, v) \in \mathcal{E}$ of $G$ is mapped to an edge $e = \{u, v\} \in \mathcal{E}$ whose relation is $\mathcal{R}_e = (s - 1) * S_u + S_v$, with the position map $P_e(u) = 1$ and $P_e(v) = 2$. 

The proposition can be trivially adapted to undirected edges, and trivially extended to heterogeneous graph with both node and edge types. Because of the proposition, we note that MPNNs on heterogeneous graphs are a special case of G-MPNN.

Several notable existing methods in the literature that can be described using our G-MPNN framework as we can see in Table 1. The proofs of how they can be described are in the appendix.
Overview. We now proceed to address limitations of existing MPNNs. In section 4, we introduce recursive hypergraphs, and then extend MPNN to MPNN-R (MPNN-Recursive) for recursive hypergraphs. Then, in the next subsection, motivated by the strong inductive capability of MPNN [26], we propose a novel instantiation of G-MPNN (Generalised-MPNN) framework, and explore the unexplored problem of inductive vertex embedding in multi-relational ordered hypergraphs. We leave exploration of G-MPNN-R as a promising future direction.

3.4 Inductive Vertex Embedding with G-MPNN

Though there exist several published works for learning on multi-relational ordered hypergraphs [66,16,24,51,24,33], none of them is an MPNN-based approach and they all are restricted to the transductive setting (i.e. assume that all vertices are present during training). Motivated by the strong inductive capability of MPNN [26,57] (and hence G-MPNN), we explore the unexplored problem of inductive learning (i.e. inductive vertex embedding) on multi-relational ordered hypergraphs.

We propose the following simple function for $M$:

$$M_t(h_v, \{ (w, h_w^t) \}_{w \in e-v}, \mathcal{R}(e, P_e), P_e) = r_{e,P_e}^t \cdot p_{e,v}^t \cdot h_v^t \cdot \prod_{w \in e-v} (p_{e,w}^t \cdot h_v^t)$$

(3)

where we use bold letters for embeddings. We use linear layers to transform relation, and position embeddings between time steps i.e. $r_{e,P_e}^{t+1} = W_{r} \cdot r_{e,P_e}^t$ and $p_{e,v}^{t+1} = W_{p} \cdot p_{e,v}^t$. For each vertex, our proposed method learns a function on the vertex embeddings of the vertex’s neighbourhood, and any available vertex features. This precisely what enables the inductiveness of the method. It is worth noting that if the hypergraph $\mathcal{H}$ is attributed i.e. all the vertices, whether seen or unseen during training, are associated with input features (such as entity descriptions), then we use Equation (3) to compute the hidden representations of unseen vertices at test time.

4 MPNN-R: Message Passing for Recursive Hypergraphs

In this section, we will first see definitions with notations used. We will then see MPNN-R to handle recursive hypergraphs [17,52].

Definition 1 (Depth $k$ powerset). For a set $S$, let us use $\mathcal{S}(S)$ to denote the powerset of $S$ i.e. $\mathcal{S}(S) := \{ \hat{S} : \emptyset \subseteq \hat{S} \subseteq S \}$. Then, the the depth $k$ powerset of $S$ is

$$2^{S,k} := \mathcal{S}\left( \bigcup_{i=0}^{k} S_i \right)$$

where $S_0 = S$, and $S_i = \mathcal{S}\left( \bigcup_{j=0}^{i-1} S_j \right)$, for $i \geq 1$

(4)

Note that $2^{S,0} = \mathcal{S}(S)$ i.e. $2^{S,0}$ is the powerset of $S$. Figure 1 shows a toy example of a depth 5 hyperedge with words as vertices. The whole hyperedge is of depth 5, pink hyperedge is of depth 4, blue of depth 3, green of depth 2, red of depth 1, and black of depth 0.

Definition 2 ($k$-Recursive hypergraph). A pair $H = (V, E)$, where $V$ is a set of $n$ vertices, and $E \subseteq \left( 2^{V,k} - \emptyset \right)$ is a set of recursive hyperedges.

Note that a hypergraph in the traditional sense is a 0-recursive hypergraph. In addition to the 5-recursive hyperedge of Figure 1, the following is a simpler example of a 1-recursive hypergraph.

Example 1. (Academic network) Let $V$ be a set a set of documents, $E = E_0 \cup E_1$ be the set of recursive hyperedges. $E_0$ contains depth 0 hyperedges of co-citation relationships (all documents cited by a document belong to a hyperedge). $E_1$ contains depth 1 hyperedges of co-authorship relationships (all documents co-authored by an author belong to a hyperedge).

An author (depth 1 hyperedge) can be a co-author of, say, 10 documents. Each of these 10 documents represents a depth 0 hyperedge connecting all its cited documents.
4.1 Laplacian-based MPNN

One of the most popular (if not the most popular), widely used examples of MPNN on graphs is the Laplacian-based convolutional neural network on graphs [36][14]. From a signal processing perspective, 2D convolutions are linear, shift-invariant (i.e. equivariant) functions on image grid data. By analogy, graph convolution on vertex-indexed signals, \( x : V \to \mathbb{R}; v \to x_v \), can be interpreted as a linear and equivariant operator with respect to (powers of) Laplacian shifts i.e. \( L^t x \) where \( L \) is the graph Laplacian [55][8]. Graph convolutional network (GCN) [30] uses a linear function of \( L \) (the graph Laplacian) for the (layer-wise) convolution operation.

4.2 Laplacian of Recursive Hypergraph

Inspired by GCN, we propose a Laplacian-based MPNN-R for recursive hypergraphs. To be able to define such an MPNN, we need a notion of Laplacian for recursive hypergraphs. Our idea crucially relies on the relationship between a Laplacian matrix and incidence matrix.

**Definition 3 (Incidence structure [2]).** A set of objects (e.g. vertices) together with certain incidence relations between these objects (e.g. hyperedges) is an incidence structure.

An incidence matrix is a rectangular matrix that shows the relationship between two classes of objects (e.g. vertices and hyperedges) in an incidence structure. It is typically represented by a rectangular \( |V| \times |E| \) matrix where \( V \) and \( E \) are two sets. Now, we state a well-known relationship between the incidence and the Laplacian matrices.

**Theorem 1.** Let \( H = (V, E) \) be an incidence structure and \( I \) be an (arbitrarily chosen) incidence matrix and Laplacian \( L \). Then \( II^T = L \).

The proof is seen in standard literature (e.g. a book [20]). The key idea of our proposed MPNN-R is to appropriately choose an incidence matrix \( I \) for the input recursive hypergraph, and then use the resulting Laplacian in a Laplacian-based MPNN.

**Incidence Matrix of Recursive Hypergraph.** Recall that in a recursive hypergraph \( H = (V, E) \), hyperedges can act as vertices in other hyperedges. Hence, we define the new vertex set \( U = V \cup E \) with the same hyperedge set \( E \). The incidence matrix \( I \) is hence a \( |U| \times |E| \) matrix where each entry \( I_{ue} \) indicates the "strength" of the membership of \( u \in U \) in the hyperedge \( e \in E \). Note that for two different hyperedges \( e_1 \in E \) and \( e_2 \in E \), the strengths might be different i.e. \( I_{ue_1} \neq I_{ue_2} \). This can be seen as a generalisation of hyperedge-dependent vertex weights [13][38] to recursive hypergraphs. Hyperedge-dependent vertex weights are known to utilise higher-order relationships in hypergraphs.

4.3 Laplacian-based MPNN-R for Recursive Hypergraph

From Theorem 1 we can now compute the Laplacian matrix of the recursive hypergraph as \( L = II^T \) (where \( I \) is obtained as above). Our MPNN-R then takes the following form for \( v \in U \):

\[
\begin{align*}
    m_v^{t+1} &= g\left(\left\{ M_t\left(h_v^t, h_u^t, e_{uv}\right)\right\}_{u \in N_v}\right), \\
    h_v^{t+1} &= U_t\left(h_v^t, m_v^{t+1}\right), \\
    M_t\left(h_v^t, h_u^t, e_{uv}\right) &= L_{vu}h_u^t, \\
    U_t\left(h_v^t, m_v^{t+1}\right) &= \sigma\left((W^t)^T m_v^{t+1}\right)
\end{align*}
\]

(5)

We define the "neighbourhood" of \( v \) as those vertices whose entries in the Laplacian matrix corresponding to row of vertex \( v \) are non-zero. It is easy to see that Laplacian-based MPNNs for graphs such as GCN [30] and hypergraphs such as HGNN [17] are examples of our MPNN-R (for special incidence matrices) because they essentially work on 0-recursive (hyper)graphs.

5 Evaluation

We evaluate the performance of the proposed methods in a number of experiments. Following standard practices of prior works, we focus on the following two most popular tasks.
We found that sum and max aggregators perform comparably to the mean aggregator. Please see the appendix for detailed ablation studies. We train our MPNN-R with cross entropy loss function on the

### Experimental Set-up

To contextualise the experimental results on the datasets, we compare against the following four baselines:

- **Multi-layer Perceptron (MLP):** We use the initial features into a simple 2-layer feed-forward neural network to predict the class labels of instances. This baseline ignores the recursive hypergraph structure.

- **HyperGraph Neural Network (HGNN [17]):** HGNN approximates the input hypergraph by introducing pairwise connections among all vertex pairs in each hyperedge [80]. This is exactly the same as a straightforward extension of GCN [56] to hypergraphs. We use all the depth 0-hyperedges to form the input hypergraph for HGNN.

- **HyperGraph Convolutional Network (HyperGCN [70]):** HyperGCN uses the mediator expansion [10] to approximate the input hypergraph into a graph. As with HGNN, we use all the depth 0-hyperedges to form the input hypergraph for HyperGCN.

- **Heterogeneous Graph Neural Network (HetGNN):** We compare against a GNN-based method on heterogeneous graphs [73]. We treat instances, depth 0-hyperedges, and depth 1-hyperedges as three different types of vertices in a heterogeneous graph. We connect an instance, and a depth 0-hyperedge if the instance belongs to the hyperedge. We connect a depth 0, and a depth 1-hyperedge if the depth 0-hyperedge belongs to the depth 1-hyperedge.

- **Heterogeneous Attention Network (HAN) [65]:** This is another GNN-based method on heterogeneous graphs that uses an attention-based formulation.

- **Metapath-Aggregated Graph Neural Network (MAGNN) [19]:** This is a recent GNN-based method on heterogeneous graphs that exploits metapaths (schemas) for neighbourhood aggregation.

### Model Details

We use a 2-layer MPNN-R of Equation 5 with ReLU as the non-linear activation function. We use 32-dimensional hidden embeddings with c-dimensional output embeddings where c is the number of classes as shown in dataset statistics table in the appendix. We set the hyperedge-dependent vertex weights to one for all vertices i.e. \( T_{ue} = 1 \) if \( u \in e \). Please see the appendix for comparison with different hyperedge-dependent vertex weights. We use the popularly-used mean aggregator to aggregate messages from the neighbourhood i.e. given a set of hidden embeddings denoted by \( h \) we use the following function in Equation 5

\[
g \left( \{ h_u \}_{u \in N_v} \right) = \frac{1}{|N_v|} \sum_{u \in N_v} h_u
\]

We found that sum and max aggregators perform comparably to the mean aggregator. Please see the appendix for detailed ablation studies. We train our MPNN-R with cross entropy loss function on the
Table 3: Results of link prediction experiments. We report MRR, Hits@1, and Hits@3 (higher is better). Please refer to section 5.2 for details.

<table>
<thead>
<tr>
<th>Method</th>
<th>WikiPeople</th>
<th>JF17K</th>
<th>M-FB15K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Hits@1</td>
<td>Hits@3</td>
</tr>
<tr>
<td>HGNN</td>
<td>0.132</td>
<td>0.111</td>
<td>0.157</td>
</tr>
<tr>
<td>HyperGCN</td>
<td>0.138</td>
<td>0.109</td>
<td>0.158</td>
</tr>
<tr>
<td>G-MPNN-sum (ours)</td>
<td>0.257</td>
<td>0.216</td>
<td>0.308</td>
</tr>
<tr>
<td>G-MPNN-mean (ours)</td>
<td>0.281</td>
<td>0.220</td>
<td>0.315</td>
</tr>
<tr>
<td>G-MPNN-max (ours)</td>
<td>0.289</td>
<td>0.218</td>
<td>0.310</td>
</tr>
</tbody>
</table>

We evaluate G-MPNN on the task of link prediction on real-world datasets. We report mean errors and their standard deviations (lower is better) of the test splits over 100 different train-test splits.

Experimental Results: We take extensive steps to avoid any kind of bias in our results. We optimise all hyperparameters of all baselines and our method using grid search. Please see the appendix for more details on the hyperparameters used. Table 2 shows the results of semi-supervised vertex classification on real-world datasets. We report mean errors and their standard deviations (lower is better) of the test splits over 100 different train-test splits.

Discussion: As we can see from Table 2, MLP is least effective on all datasets. This shows that the input recursive hypergraph is informative in classifying vertices. We can also observe that HetGNN that uses both depth 0 and depth 1-hyperedges is consistently superior to HGNN and HyperGCN both of which use only the depth 0-hyperedges. Finally, our proposed method MPNN-R is able to consistently outperform all the baselines including HetGNN. We believe this is because of the principled integration of depth 0, depth 1-hyperedges, and also relationships among them. Finally we performed a statistical test to validate the significance of the results. Based on a Welch t-test, the p-value on DBLP is 0.35, and a small p-value on the other three (less than 0.0005).

5.2 G-MPNN: Link Prediction

We evaluate G-MPNN on the task of link prediction on N-ary relational facts. Since the inductive version (embedding unseen entities at test time) is an unexplored task, we create datasets from existing (transductive) datasets. Details of the dataset construction and statistics are in the appendix.

Inductive Vertex Embedding: Since the vertices of the hypergraph are not attributed, we make a (reasonable) assumption that the neighbourhood of an unseen vertex is informative, and use Equation 3 only for the neighboring vertices seen during training. In other words, if seen(e) is the set of vertices in hyperedge e seen during training then for an unseen vertex v ∈ e, we modify Equation 3 to

\[ M_t\left(h^t_v, \{ (w, h^t_w) \}_{w \in e - v}, \mathcal{R}(e, P_e), P_e \right) = r^t_e, P_e \right) = r^t_e, P_e * \prod_{w \in \text{seen}(e)} (p^t_e, w * h^t_w) \] (6)

Objective Function: As the input hypergraph has only positive hyperedges, we produce a set of \( \eta \) negative hyperedges through a standard contrastive generation procedure [6]. The score, \( \phi(e) \), of a hyperedge, \( e \), after obtaining G-MPNN vertex hidden representations, is computed using generalised bilinear scoring for N-ary facts [10] i.e.

\[ \phi(e) = r^T_e, P_e \right) = r^T_e, P_e \right) = r^T_e, P_e \prod_{w \in e} (p^T_e, w * h^T_w) \] where T is the total number of G-MPNN time steps. We use a combination of softmax and negative log likelihood loss which is shown to be effective for link prediction [34].

\[ \mathcal{L} = \sum_{e \in \mathcal{E}} -\log \left( \frac{\exp(\phi(e))}{\exp(\phi(e)) + \sum_{e' \in \text{neg}(e)} \exp(\phi(e'))} \right) \]

Experimental Set-up: To contextualise the experimental results on the datasets, we compare against two baselines - HGNN [17] and HyperGCN [20] adapted to the problem of link prediction. Both HGNN and HyperGCN were originally proposed for vertex-level tasks on hypergraphs but we adapt them to link prediction task by removing the relational and ordering information from the multi-relational ordered hypergraph. We use the hidden representations obtained from these models as embeddings for the scoring function \( \phi(e) \).
Model Details: We use a 2-layer G-MPNN with ReLU as the non-linear activation function. We use 200-dimensional hidden units in both the layers. We use three popular forms for the differentiable function $g$ viz., sum, mean, and max \cite{62} which we call G-MPNN-sum, G-MPNN-mean, and G-MPNN-max respectively. All models are implemented in PyTorch \cite{50} using Adam \cite{35}.

Metrics for Evaluation: We rely on two popular ranking metrics viz. Hits@$k$, and Mean Reciprocal Rank (MRR). Both these metrics rank a test hyperedge $e$ within a set of corrupted (negative) hyperedges. Since the number of positions is small (less than 8 for all datasets), for each $(e, P_e)$ pair and for each position, we generate $|V| - 1$ corrupted hyperedges by replacing the vertex $v \in e$ at position $i$ with each of the entities in $V - v$. We rank the positive $e$ based on score $\phi(e)$. We compute MRR as mean of the sum (over all positions) of the reciprocals of the rank of $e$. Hits@$k$ is the proportion of hyperedges that rank among top $k$ in corrupted sets.

Experimental Results and Discussion: The results of link prediction are shown in Table 3. Firstly, our proposed methods are more effective than HGNN and HyperGCN. We believe this is because the two baselines do not exploit the positional and relational information in the hypergraph. We have conducted ablation studies by removing positional and relational information and the results are in the appendix. We have also conducted experiments on transductive datasets and the results (results in the appendix).

6 Conclusion

In this work, we have unified existing MPNN approaches proposed on a wide range of networks such as multi-relational graphs, hypergraphs, heterogeneous graphs, etc. Our unified framework, G-MPNN, has attractive properties including strong inductive capability on multi-relational ordered hypergraphs. We have also proposed a novel framework, MPNN-R, on recursive hypergraphs. Future works include exploiting relational and positional information in recursive hypergraphs (see Figure 1) for natural language processing tasks. Another interesting direction is to investigate and extend the recent subgraph reasoning methods \cite{74,57} for inductive vertex embedding for multi-relational ordered hypergraphs.

Broader Impact

Message Passing Neural Networks (MPNNs) are a framework for deep learning on graph structured data. Graph structures are universal and very generic structures commonly seen in various forms in computer vision, natural language processing, recommender systems, traffic prediction, generative models, and many more. Graphs can have many variations such as multi-relational, heterogeneous, hypergraphs, etc.

Our research in this paper unifies several existing MPNN methods on these variations. While we show how our research could be used for academic networks, and factual knowledge, it opens up many more possibilities in natural language processing (NLP). We see opportunities for research applying our work for beneficial puroposes, such as investigating whether we could improve performance of NLP tasks such as machine reading comprehension, relation extraction, machine translation, and many more.

Potentially hazardous applications include trying to predict criminality or credit from social networks. Such applications may reproduce and exacerbate bias and readers of the paper should be aware that the presented model should not applied naively to such tasks.

Acknowledgement

This work is supported by the Ministry of Human Resource Development (Government of India). We give credits to Google Images and Wikipedia for the pictures used in the paper.
References


[49] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In Proceedings of the Thirty-Fourth Conference on Association for the Advancement of Artificial Intelligence (AAAI), 2020.


