A Hierarchical Entity Graph Convolutional Network for Relation Extraction across Documents

Tapas Nayak*
Department of Computer Science
Indian Institute of Technology Kharagpur
tnk02.05@gmail.com

Hwee Tou Ng
Department of Computer Science
National University of Singapore
nght@comp.nus.edu.sg

Abstract

Distantly supervised datasets for relation extraction mostly focus on sentence-level extraction, and they cover very few relations. In this work, we propose cross-document relation extraction, where the two entities of a relation tuple appear in two different documents that are connected via a chain of common entities. Following this idea, we create a dataset for two-hop relation extraction, where each chain contains exactly two documents. Our proposed dataset covers a higher number of relations than the publicly available sentence-level datasets. We also propose a hierarchical entity graph convolutional network (HEGCN) model for this task that improves performance by 1.1% F1 score on our two-hop relation extraction dataset, compared to some strong neural baselines.

1 Introduction

The idea of distant supervision (Mintz et al., 2009) eliminates the need for manual annotation for obtaining training data for relation extraction. Previously, this idea is used mostly to create sentence-level datasets. However, the assumption of distant supervision, that the two entities of a tuple must appear in the same sentence, is overly strict. We may not find an adequate number of evidence sentences for many relations as both entities do not appear in the same sentence. The relation extraction models built on such data can find relations only for a small number of relations and the relations of most knowledge bases (KBs) will be out of the reach of such models.

To address this issue, we propose a multi-hop relation extraction task where the subject and object entities of a tuple can appear in two different documents, and these two documents are connected via some common entities. We can create a chain of entities from the subject entity to the object entity of a tuple via the common entities across multiple documents. Each link in this chain represents a relation between the entities located at the endpoints of the link. We can determine the relation between the subject and object entities of a tuple by following this chain of relations. This approach can give training instances for more relations than sentence-level distant supervision. Following the proposed multi-hop approach, we create a two-hop relation extraction dataset for the task. Each instance of this dataset has two documents, where the first document contains the subject entity and the second document contains the object entity of a tuple. These two documents are connected via at least one common entity. This idea can be extended to create an N-hop dataset.

We also propose a hierarchical entity graph convolutional network (HEGCN) model for the task. Our proposed model has two levels of graph convolutional networks (GCNs). The first-level GCN of the hierarchy is applied to the entity mention level graph of every document to capture the relations among the entity mentions within a document. The second-level GCN of the hierarchy is applied on a unified entity-level graph, which is built using all the unique entities present in the document chain. This entity-level graph can be built on the document chain of any length and it can capture the relations among the entities across the multiple documents in the chain. Our proposed HEGCN model improves the performance on our two-hop dataset. To summarize, the following are the contributions of this paper:

(1) We propose a multi-hop relation extraction task and create a two-hop dataset. This dataset has more relations than other popular distantly supervised sentence-level or document-level relation extraction datasets.

* This work was done when the first author was a PhD student at the National University of Singapore.

Proceedings of Recent Advances in Natural Language Processing, pages 1022–1030
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_115

(2) We propose a novel hierarchical entity graph convolutional network (HEGCN) for multi-hop relation extraction. Our proposed model improves the F1 score by 1.1% on our two-hop dataset, compared to strong neural baselines[1].

2 Task Formalization

Multi-hop relation extraction can be defined as follows. Consider two entities, a subject entity e_s and an object entity e_o, and a chain of documents $D = \{D_s \rightarrow D_1 \rightarrow D_2 \rightarrow ... \rightarrow D_n \rightarrow D_o\}$ where $e_s \in D_s$ and $e_o \in D_o$. There exists a chain of entities $e_s \rightarrow c_1 \rightarrow c_2 \rightarrow ... \rightarrow c_{n+1} \rightarrow e_o$ where $c_1 \in \{D_s, D_1\}$, $c_2 \in \{D_1, D_2\}$, ..., $c_{n+1} \in \{D_n, D_o\}$. The task is to find the relation between e_s and e_o from a pre-defined set of relations $R \cup \{None\}$, where R is the set of relations and *None* indicates that none of the relations in R holds between e_s and e_o. A simpler version of this task is two-hop relation extraction where D_s and D_o are directly connected by at least one common entity. In this paper, we focus on two-hop relation extraction.

3 Related Work

3.1 Relation Extraction Datasets

Distantly supervised datasets are very popular for relation extraction (Nayak et al., 2021). Riedel et al. (2010) (NYT10) and Hoffmann et al. (2011) (NYT11) mapped Freebase tuples to New York Times (NYT) articles to obtain such datasets. The NYT10 and NYT11 datasets have been used extensively by researchers for relation extraction. TACRED (Zhang et al., 2017) is another dataset created from the TAC KBP evaluations. FewRel 2.0 (Gao et al., 2019) is a few-shot relation extraction dataset. All these datasets are created at the sentence level. DocRED (Yao et al., 2019) is a document-level relation extraction dataset created using Wikipedia articles and Wikidata items. To the best of our knowledge, there does not exist any relation extraction dataset which involves multiple documents.

3.2 Relation Extraction Models

Neural models have performed well on distantly supervised datasets for relation extraction. Zeng et al. (2014, 2015) used convolutional network with max-pooling on word embeddings for this task, whereas Shen and Huang (2016); Jat et al. (2017); Nayak and Ng (2019) used word-level attention model for single-instance sentence-level relation extraction. Lin et al. (2016); Vashishth et al. (2018); Ye and Ling (2019) used neural networks in a multi-instance setting to find a relation from a bag of independent sentences. Recently, graph convolutional network-based (GCN) (Kipf and Welling, 2017) models have become popular for many NLP tasks. These models work on non-linear graph structures. Zhang et al. (2018); Vashishth et al. (2018); Guo et al. (2019); Zeng et al. (2020) used graph convolution networks for relation extraction. They consider each token in a sentence as a node in the graph and use a syntactic dependency tree to create a graph structure among the nodes. Recently, neural joint extraction approaches (Takanobu et al., 2019; Nayak and Ng, 2020) were proposed for this task.

3.3 Multi-hop QA versus Multi-hop RE

Welbl et al. (2018) proposed a multi-hop QA dataset (WikiHop) where the answer can only be found using more than one document. Several neural models have been proposed (Song et al., 2018; Cao et al., 2019; De Cao et al., 2019; Kundu et al., 2019) to solve this task. We have created a two-hop relation extraction dataset (THRED) from this WikiHop dataset. The major difference between these two datasets is that THRED contains many *None* relations, whereas in the WikiHop dataset, every instance has a correct answer. Extracting the *None* relation is challenging, since *None* occurs when no relations in R exist. When the number of relations in R increases, it becomes more difficult to predict the relations. As such, we believe the multi-hop RE task is more challenging than the multi-hop QA task.

4 Dataset Construction

We create a two-hop relation extraction dataset from a multi-hop question-answering (QA) dataset WikiHop (Welbl et al., 2018). Welbl et al. (2018) defined the multi-hop QA task as follows: Given a set of supporting documents D_s and a set of candidate answers C_a which are mentioned in D_s, the goal is to find the correct answer $a^* \in C_a$ for a question by drawing on the supporting documents. They used Wikipedia articles and Wikidata (Vrandečić and Krötzsch, 2014) tuples for creating this dataset. Each positive tuple (e_s, e_o, r_p) in

[1]The source code and data for this paper are available at https://github.com/nusnlp/MHRE.git

Wikidata has two entities, a subject entity e_s and an object entity e_o, and a positive relation r_p between the subject and object entity. The questions are created by combining the subject entity e_s and the relation r_p, and the object entity e_o is the correct answer a^* for a given question. The other candidate answers are carefully chosen from Wikidata entities so that they have a similar type as the correct answer. The supporting documents are chosen in such a way that at least two documents are needed to find the correct answer. This means the subject entity e_s and the object entity e_o do not appear in the same document. They used a bipartite graph partition technique to create the dataset. In this bipartite graph, vertices on one side correspond to Wikidata entities, and vertices on the other side correspond to Wikipedia articles. An edge is created between an entity vertex and a document vertex if this document contains the entity. As we traverse the graph starting from vertex e_s, it visits many document vertices and entity vertices. This constitutes the supporting document set and candidate answer set. If the candidate answer set does not contain the object entity e_o which is the correct answer, this instance is discarded. They also limited the length of the traversal to three documents. Welbl et al. (2018) only released the supporting documents, questions, and candidate answers for their dataset. They did not release the connecting entities.

We convert this WikiHop dataset into a two-hop relation extraction dataset. The subject entities and the candidate entities can be easily found in the documents using string matching. We use a named entity recognizer from spaCy[2] to find the other entities in the documents and these entities can link these documents. We find that most of the WikiHop question-answer instances are two-hop instances. That means for most of the instances of WikiHop dataset, there is at least one document pair in the supporting document set where the first document of the pair contains the subject entity and the second document of the pair contains the correct answer, and these two documents in the pair are directly connected via some third entity. To simplify the multi-hop relation extraction task, we fix the hop count at 2. For every instance of the WikiHop dataset, we can easily find the subject entity e_s and the positive relation r_p from the question. The correct answer a^* is the object entity of a positive tuple. (e_s, a^*, r_p) is the positive tuple for relation extraction. For any other candidate answer $e_w \in C_a - \{a^*\}$, the entity pair (e_s, e_w) is considered as a *None* tuple if there exists no relation among the four pairs (e_s, e_w), (e_w, e_s), (e_w, e_o), and (e_o, e_w) in Wikidata. We check for the no relation condition for these four entity pairs involving e_w, e_s, and e_o to reduce the distant supervision noise in the dataset for *None* tuples. We create a *None* candidate set C_n with each $e_w \in C_a - \{a^*\}$. We first find all possible pairs of documents from the supporting document set D_s such that the first document of the pair contains the subject entity e_s and the second document of the pair contains either the entity a^* or one of the entities from C_n. We discard those pairs of documents that do not contain any common entity. The document pairs where the second document contains the entity a^* are considered as a document chain for the positive tuple (e_s, a^*, r_p) where $r_p \in R$. All other document pairs where the second document contains an entity from the set C_n are considered as a document chain for *None* tuple $(e_s, e_w, None)$ where $e_w \in C_n$. In this way, using distant supervision, we can create a dataset for two-hop relation extraction. Each instance of this dataset has a chain of documents $D = \{D_s \rightarrow D_o\}$ of length 2 that is the textual source of a tuple (e_s, e_o, r). The document D_s contains the subject entity e_s and the document D_o contains the object entity e_o. The two documents are connected with at least one common entity c. There exists at least one entity chain $e_s \rightarrow c \rightarrow e_o$ in the document chain. The goal is to find the relation r between e_s and e_o from the set $R \cup \{None\}$. We refer to this two-hop dataset as THRED (two-hop relation extraction dataset) in the remaining sections of this paper. We manually checked 100 randomly selected positive samples and 100 randomly selected negative samples, and found that 76% of the selected positive samples and 82% of the selected negative samples are accurate.

4.1 Dataset Statistics

The training, validation, and test data of the WikiHop dataset are created using distant supervision, but the validation and test data are manually verified. WikiHop test data is blind and not released. So we use their validation data to create the test data for our task and use their training data for our training and validation purposes. We include the statistics of our two-hop relation extraction dataset

[2]https://spacy.io/

Question	located_in_administrative_entity Zoo Lake
Candidates	Gauteng, Tanzania
Answer	Gauteng
Doc1	**Zoo Lake** is a popular lake and public park in **Johannesburg** , **South Africa** . It is part of the **Hermann Eckstein Park** and is opposite the **Johannesburg Zoo** . The **Zoo Lake** consists of two dams , an upper feeder dam , and a larger lower dam , both constructed in natural marshland watered by the **Parktown Spruit** .
Doc2	**Johannesburg** is the largest city in **South Africa** and is one of the 50 largest urban areas in the world . It is the provincial capital of **Gauteng** , which is the wealthiest province in **South Africa** .
Doc3	**Mozambique** is a country in **Southeast Africa** bordered by the **Indian Ocean** to the east , **Tanzania** to the north , **Malawi** and **Zambia** to the northwest , **Zimbabwe** to the west , and **Swaziland** and **South Africa** to the southwest .

Table 1: A multi-hop question-answer instance from the WikiHop dataset. The tuple (Doc1, *Zoo Lake*, Doc2, *Gauteng*, *located_in_administrative_entity*) constitutes a positive instance in the THRED dataset. The tuple (Doc1, *Zoo Lake*, Doc3, *Tanzania*, *None*) constitutes a negative instance in the THRED dataset.

	Train	Test
#Positive relations	218	72
#Document chains	143,906	5,320
#Positive instances	40,247	1,672
#Positive entity pairs	21,490	618
#None instances	197,731	7,806

Table 2: Statistics of the THRED dataset.

in Table 2. We include the statistics on the number of common entities present in the two documents of a chain in Table 3. We split the training data randomly, with 90% for training and 10% for validation. From Table 2, we see that the dataset contains a much higher number of *None* tuples than the positive tuples. So we randomly select *None* tuples so that the number of *None* tuples is the same as the number of positive tuples for training and validation. For evaluation, we consider the entire test dataset. From Table 4, we see that our THRED dataset contains more relations than any other distantly supervised relation extraction datasets such as the New York Times (Riedel et al., 2010; Hoffmann et al., 2011) or DocRED (Yao et al., 2019).

5 Proposed HEGCN Model

We propose a hierarchical entity graph convolutional network (HEGCN) for multi-hop relation extraction. We encode the documents in a document chain using a bi-directional long short-term memory (BiLSTM) layer (Hochreiter and Schmidhuber, 1997). On top of the BiLSTM layer, we use two graph convolutional networks (GCN), one after another in a hierarchy. In the first level of the GCN hierarchy, we construct a separate entity mention graph on each document of the chain using all the entities mentioned in that document. Each mention of an entity in a document is considered as a separate node in the graph. We use a graph convolutional network (GCN) to represent the entity mention graph of each document to capture the relations among the entity mentions in the document. We then construct a unified entity-level graph across all the documents in the chain. Each node of this entity-level graph represents a unique entity in the document chain. Each common entity between two documents in the chain is represented by a single node in the graph. We use a GCN to represent this entity-level graph to capture the relations among the entities across the documents. We concatenate the representations of the nodes of the subject entity and object entity and pass it to a feed-forward layer with softmax for relation classification.

	#Document chains	
#Common entities	Train	Test
1	92,140	3,615
2	36,275	1,161
3	10,824	374
4	3,170	113
$\geq$5	1,497	57

Table 3: Statistics of the common entities in the THRED dataset.

Dataset	$\|R\|$	Dataset	$\|R\|$
NYT10	53	NYT11	24
TACRED	41	DocRED	96
FewRel 2.0	100	**THRED**	**218**

Table 4: The number of relations in various relation extraction datasets. R is the set of positive relations.

5.1 Documents Encoding Layer

We use two types of embedding vectors: (1) word embedding vector $\mathbf{w} \in \mathbb{R}^{d_w}$ (2) entity token indicator embedding vector $\mathbf{z} \in \mathbb{R}^{d_z}$, which indicates if a word belongs to the subject entity, object entity, or common entities. The subject and object entities are assigned the embedding index of 2 and 3, respectively. The common entities in the document chain are assigned embedding index in an

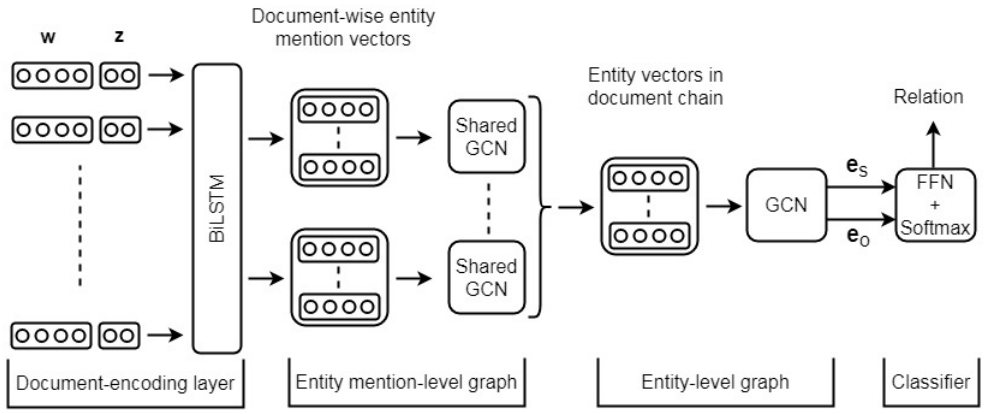

Figure 1: The architecture of our proposed HEGCN model. GCN in entity mention-level graph is shared across the documents in a chain. This diagram is for document chain of length 2.

increasing order starting from index 4. The same entities present in two documents in the chain get the same embedding index. Embedding index 0 is used for padding and 1 is used for all other tokens in the documents. A document is represented using a sequence of vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ where $\mathbf{x}_t = \mathbf{w}_t \| \mathbf{z}_t$. $\|$ represents the concatenation of vectors and n is the document length. We concatenate all documents in a chain sequentially by using a document separator token. These token vectors are passed to a BiLSTM layer to capture the interaction among the documents in a chain. $\overrightarrow{\mathbf{h}_t} \in \mathbb{R}^{(d_w+d_z)}$ and $\overleftarrow{\mathbf{h}_t} \in \mathbb{R}^{(d_w+d_z)}$ are the output at the tth step of the forward LSTM and backward LSTM respectively. We concatenate them to obtain the tth BiLSTM output $\mathbf{h}_t \in \mathbb{R}^{2(d_w+d_z)}$.

5.2 Hierarchical Entity Graph Convolutional Layers

Kipf and Welling (2017) proposed graph convolutional networks (GCN) which work on graph structures. Here, we describe the GCN which is used in our model. We represent a graph $\mathcal{G}$ with m nodes using an adjacency matrix $\mathbf{A}$ of size $m \times m$. If there is an edge between node i and node j, then $A_{ij} = A_{ji} = 1$. We also add self loops, $A_{ii} = 1$, in the graph $\mathcal{G}$. We normalize the adjacency matrix $\mathbf{A}$ by using symmetric normalization proposed by Kipf and Welling (2017). A diagonal node degree matrix $\mathbf{D}$ of size $m \times m$ is used in the normalization of $\mathbf{A}$. $\deg(v_i)$ is the number of edges that are connected to the node v_i in $\mathcal{G}$ and $\hat{\mathbf{A}}$ is the corresponding normalized adjacency matrix of $\mathcal{G}$. Each node of the graph receives the hidden representation of its neighboring nodes from the $(l-1)$th layer and uses the following operation to update its own hidden representation.

$$D_{ij}^{-\frac{1}{2}} = \begin{cases} \frac{1}{\sqrt{\deg(v_i)}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

$$\mathbf{g}_i^l = \text{ReLU}(\sum_{j=1}^{m} \hat{A}_{ij}\mathbf{W}^l\mathbf{g}_j^{l-1})$$

$\mathbf{W}^l$ is the trainable weight matrix of the lth layer of the GCN, $\mathbf{g}_i^l$ is the representation of the ith node of the graph at the lth layer. If $\mathbf{g}_i^l$ has the dimension of d_g, then the dimension of the weight matrix $\mathbf{W}^l$ is $d_g \times d_g$. $\mathbf{g}_i^0$ is the initial input to the GCN.

5.2.1 Entity Mention Graph Layer

We construct an entity mention graph (EMG) for each document in the chain on top of the document encoding layer. An entity string may appear at multiple locations in a document and each appearance is considered as an entity mention. We add a node in the graph for each entity mention. We connect two entity mention nodes if they appear in the same sentence (EMG type 1 edge). We assume that since they appear in the same sentence, there may exist some relation between them. We also connect two entity mention nodes if the strings of the two entity mentions are identical (EMG type 2 edge). Let $e_1, \ldots, e_l$ be the sequence of entity mention nodes listed in the order of their appearance in a document. We connect nodes e_i and e_{i+1} $(1 \leq i < l)$ with an edge (EMG type 3 edge). EMG type 3 edges create a linear chain of the entity mentions and ensure that the graph is connected. We

use a graph convolutional network on this graph topology to capture the relations among the entity mentions in a document.

We obtain the initial representations of the entity mention nodes from the hidden representations of the document encoding layer. We concatenate the hidden vector of the first token of an entity mention, the hidden vector of its last token, and a context vector to obtain the entity mention node representation. The context vector is obtained using an attention mechanism on the tokens of the sentence in which the entity mention appears.

$$\mathbf{p} = \mathbf{h}_b \parallel \mathbf{h}_e, \quad s_t = \tanh(\mathbf{p}^T \mathbf{W})\mathbf{h}_t$$
$$\mathbf{a} = \text{softmax}([s_1 s_2 \dots s_k]^T)$$
$$\mathbf{c} = \sum_{t=1}^{k} \text{a}_t \mathbf{h}_t, \quad \mathbf{q} = \mathbf{p} \parallel \mathbf{c}$$

$\mathbf{h}_b \in \mathbb{R}^{2(d_w+d_z)}$ and $\mathbf{h}_e \in \mathbb{R}^{2(d_w+d_z)}$ are the hidden vectors from the document encoding layer of the first and last token of an entity mention. $\mathbf{W} \in \mathbb{R}^{4(d_w+d_z)\times 2(d_w+d_z)}$ is a trainable weight matrix, $\mathbf{h}_t \in \mathbb{R}^{2(d_w+d_z)}$ is the hidden vector of the tth token of the sentence in which the entity mention is located, and a_t is the normalized attention score for the tth token with respect to the entity mention. k is the length of the sentence in which the entity mention is located, and $\mathbf{c} \in \mathbb{R}^{2(d_w+d_z)}$ is the context vector. The entity mention node vector $\mathbf{q} \in \mathbb{R}^{6(d_w+d_z)}$ of the ith node in the graph is passed to the GCN as $\mathbf{g}_i^0$. The parameters of this GCN are shared across the documents in a chain. This layer of the model is referred to as entity mention-level graph convolutional network or EMGCN.

5.2.2 Entity Graph Layer

We construct a unified entity graph (EG) on top of the entity mention graphs. First, we construct an entity graph for each document, where each unique entity string is represented as an entity node in the graph. We add an edge between two entity nodes if the strings of the two entities appear together in at least one sentence in the document (EG type 1 edge). We also form a sequence of entity nodes based on the order of appearance of the entities in a document, where only the first occurrence of multiple occurrences of an entity is kept in the sequence. We connect two consecutive entity nodes in the sequence with an edge (EG type 2 edge). This ensures that the entire entity graph remains connected.

We construct one entity graph for each document in the document chain. We unify the entity graphs of multiple documents by merging the nodes of common entities between them. The unified entity graph contains all the nodes from the multiple entity graphs, but the common entity nodes which appear in two entity graphs are merged into one node in the unified graph. There is an edge between two entity nodes in the unified entity graph if there exists an edge between them in any of the entity graphs of the documents.

We obtain the initial representations of the entity nodes from the GCN outputs of the entity mention graphs. For the common entities between two documents, we average the GCN outputs of the entity mention nodes that have an identical string as the entity from the entity mention graphs of the two documents. For other entity nodes that appear only in one document, we average the GCN outputs of the entity mention nodes that have an identical string as the entity from the entity mention graph of that document. Each entity vector is passed to another graph convolutional network as $\mathbf{g}_i^0$ which represents the initial representation of the ith entity node in the unified entity graph. We use a graph convolutional network on this graph topology to capture the relations among the entities across the documents in the document chain. This layer of the model is referred to as entity-level graph convolutional network or EGCN.

5.3 Relation Classifier

We concatenate the EGCN outputs of the nodes corresponding to the subject entity $\mathbf{e}_s \in \mathbb{R}^{6(d_w+d_z)}$ and object entity $\mathbf{e}_o \in \mathbb{R}^{6(d_w+d_z)}$, and pass the concatenated vector to a feed-forward network (FFN) with softmax to predict the normalized probabilities for the relation labels.

$$\mathbf{r} = \text{softmax}(\mathbf{W}_r(\mathbf{e}_s \parallel \mathbf{e}_o) + \mathbf{b}_r)$$

$\mathbf{W}_r \in \mathbb{R}^{(|R|+1)\times 12(d_w+d_z)}$ is the weight matrix, $\mathbf{b}_r \in \mathbb{R}^{|R|+1}$ is the bias vector of the FFN, and $\mathbf{r}$ is the vector of normalized probabilities of relation labels.

6 Experiments

6.1 Baselines

We implement four neural baseline models for comparison with our proposed HEGCN model. Similar to our proposed model, we represent the tokens in the documents using pre-trained word embedding

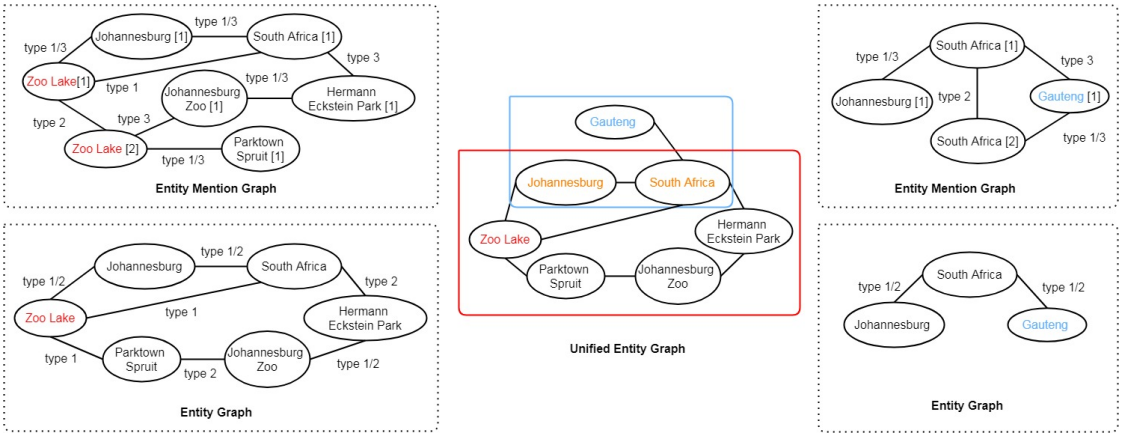

Figure 2: The graph construction process for the positive instance in Table 1. The entity mention graph and entity graph on the left are for Doc1. The entity mention graph and entity graph on the right are for Doc2. The numbers in square brackets ([x]) in the entity mention graph are used to distinguish the entity mentions with identical string. Type x/y means this edge can be of both type x and type y. The 'EMG' and 'EG' prefixes are omitted from the labels of the edges in the entity mention graph and entity graph respectively. The unified entity graph is shown in the middle. Nodes in the red box are part of the entity graph of the document containing the subject entity Zoo Lake. Nodes in the blue box are part of the entity graph of the document containing the object entity Gauteng. Common entities are marked in orange color.

vectors and entity token indicator vectors. We use a document separator token when concatenating the vectors of two documents in a chain.

(1) CNN: We apply the convolution operation on the sequence of token vectors with different kernel sizes. A max-pooling operation is applied to choose the features from the outputs of the convolution operation. This feature vector is passed to a feed-forward layer with softmax to classify the relation.

(2) BiLSTM: The token vectors of the document chain are passed to a BiLSTM layer to encode its meaning. We obtain the entity mention vectors of the subject entity and the object entity by concatenating the hidden vectors of their first and last token. We average the entity mention tokens of the corresponding entity to obtain the representation of the subject entity and the object entity. These two vectors are concatenated and passed to a feed-forward layer with softmax to find the relation between them.

(3) BiLSTM_CNN: This is a combination of the BiLSTM and CNN model described above. The token vectors of the documents are passed to a BiLSTM layer and then we use the convolution operation with max-pooling with different convolutional kernel sizes on the hidden vectors of the BiLSTM layer. The feature vector obtained from the max-pooling operation is passed to a feed-forward layer with softmax to classify the relation.

(4) LinkPath: This model uses the explicit paths (Kundu et al., 2019) from the subject entity e_s to the object entity e_o via the common entities to find the relation. As we consider only two-hop relations, each path from e_s to e_o will be of the form $e_s \rightarrow c \rightarrow e_o$, where c is a common entity. Since there can be multiple common entities between two documents and these common entities as well as the subject and object entities can appear multiple times in the two documents, there exist multiple paths from e_s to e_o. Each path is formed with four entity mentions: (i) entity mentions of the subject entity and common entity in the first document. (ii) entity mentions of the common entity and object entity in the second document. We concatenate the BiLSTM hidden vectors of the start and end token of an entity mention to obtain its representation. Each path is constructed by concatenating all the four entity mentions of the path. This can be extended from two-hop to multi-hop relations by using a recurrent neural network that takes the path entity mentions as input, and outputs the hidden representation of the path. We average the vector representations of all the paths and pass it to a feed-forward layer with softmax to find the relation.

6.2 Parameter Settings

We use GloVe (Pennington et al., 2014) word embeddings of dimension d_w which is set to 300 in

our experiments, and update the embeddings during training. We set the dimension d_z to be 20 for the entity token indicator embedding vectors. The hidden vector dimension of the forward and backward LSTM is set at 320. The dimension of BiLSTM output is 640. We use 500 different convolution filters with kernel width of 3, 4, and 5 for feature extraction. We use one convolutional layer in both entity mention-level GCN and entity-level GCN in our final model. Dropout layers (Srivastava et al., 2014) are used in our network with a dropout rate of 0.5 to avoid overfitting. We train our models with a mini-batch size of 32 and use negative log-likelihood as our objective function. We optimize the network parameters using the Adagrad optimizer (Duchi et al., 2011). For evaluation, we use precision, recall, and F1 score. We do not include the *None* relation in the evaluation. A confidence threshold that achieves the highest F1 score on the validation dataset is used to decide if the relation of a test instance belongs to the set of relations R or *None*.

6.3 Experimental Results

We include the median of five runs of the models on the THRED dataset in Table 5. We see that adding a BiLSTM in the document encoding layer improves the performance by close to 5% in F1 score. The BiLSTM, BiLSTM_CNN, and LinkPath models achieve similar F1 scores. When we add our proposed hierarchical entity graph convolutional layer on top of the BiLSTM layer, we get another 1.1% F1 score improvement over the next best BiLSTM model. We perform a statistical significance test using bootstrap resampling to compare each baseline and our HEGCN model, and have ascertained that the higher F1 score achieved by our model is statistically significant ($p < 0.001$).

Model	Prec.	Rec.	F1
CNN	0.602	0.655	0.628
BiLSTM	0.682	0.668	0.675
BiLSTM_CNN	0.654	0.696	0.674
LinkPath	0.682	0.666	0.674
HEGCN	0.674	0.699	**0.686**

Table 5: Performance comparison of the models on the THRED dataset. We report the median of 5 runs.

6.4 Ablation Studies

We include the performance of our HEGCN model with different numbers of convolutional layers in the entity mention-level GCN (EMGCN) and entity-level GCN (EGCN) in Table 6. When we increase the number of layers in either GCN, the performance of the model drops. We finally use only one convolutional layer in both EMGCN and EGCN.

L1	L2	Prec.	Rec.	F1
1	1	0.674	0.699	**0.686**
2	1	0.709	0.650	0.678
2	2	0.682	0.663	0.673
3	1	0.671	0.635	0.653
3	2	0.673	0.667	0.670
3	3	0.623	0.651	0.637

Table 6: The ablation study of the HEGCN model with different numbers of convolutional layers (L1 and L2) in EMGCN and EGCN.

In Table 7, we include the ablation study of the different types of edges in EMGCN and EGCN. Removing any type of edges reduces the F1 score.

Model	Prec.	Rec.	F1
HEGCN	0.674	0.699	**0.686**
– EMG type 1	0.679	0.689	0.684
– EMG type 2	0.698	0.662	0.680
– EMG type 3	0.666	0.693	0.679
– EG type 1	0.704	0.659	0.681
– EG type 2	0.674	0.691	0.683

Table 7: The ablation study of the different types of edges in our HEGCN model.

7 Conclusion

In this paper, we propose how the idea of distant supervision can be extended from sentence-level extraction to multi-hop extraction to cover more relations. We propose a general approach to create multi-hop relation extraction datasets. Following this approach, we create a two-hop relation extraction dataset that covers a higher number of relations from knowledge bases than other distantly supervised relation extraction datasets. We also propose a hierarchical entity graph convolutional network for this task. The two levels of GCN in our model help to capture the relation cues within documents and across documents. Our proposed model improves the F1 score by 1.1% on our two-hop dataset, compared to a strong neural baseline, and it can be readily extended to N-hop datasets.

References

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *NAACL*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In *EMNLP and IJCNLP*.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In *AKBC*.

Thomas Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Souvik Kundu, Tushar Khot, Ashish Sabharwal, and Peter Clark. 2019. Exploiting explicit paths for multi-hop reading comprehension. In *ACL*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL and IJCNLP*.

Tapas Nayak, Navonil Majumder, Pawan Goyal, and Soujanya Poria. 2021. Deep neural approaches to relation triplets extraction: A comprehensive survey. *Cognitive Computing*.

Tapas Nayak and Hwee Tou Ng. 2019. Effective attention modeling for neural relation extraction. In *CoNLL*.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML and KDD*.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *COLING*.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *AAAI*.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. RESIDE: Improving distantly-supervised neural relation extraction using side information. In *EMNLP*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of Association for Computing Machinery*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *ACL*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Distant supervision relation extraction with intra-bag and inter-bag attentions. In *NAACL*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *EMNLP*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.